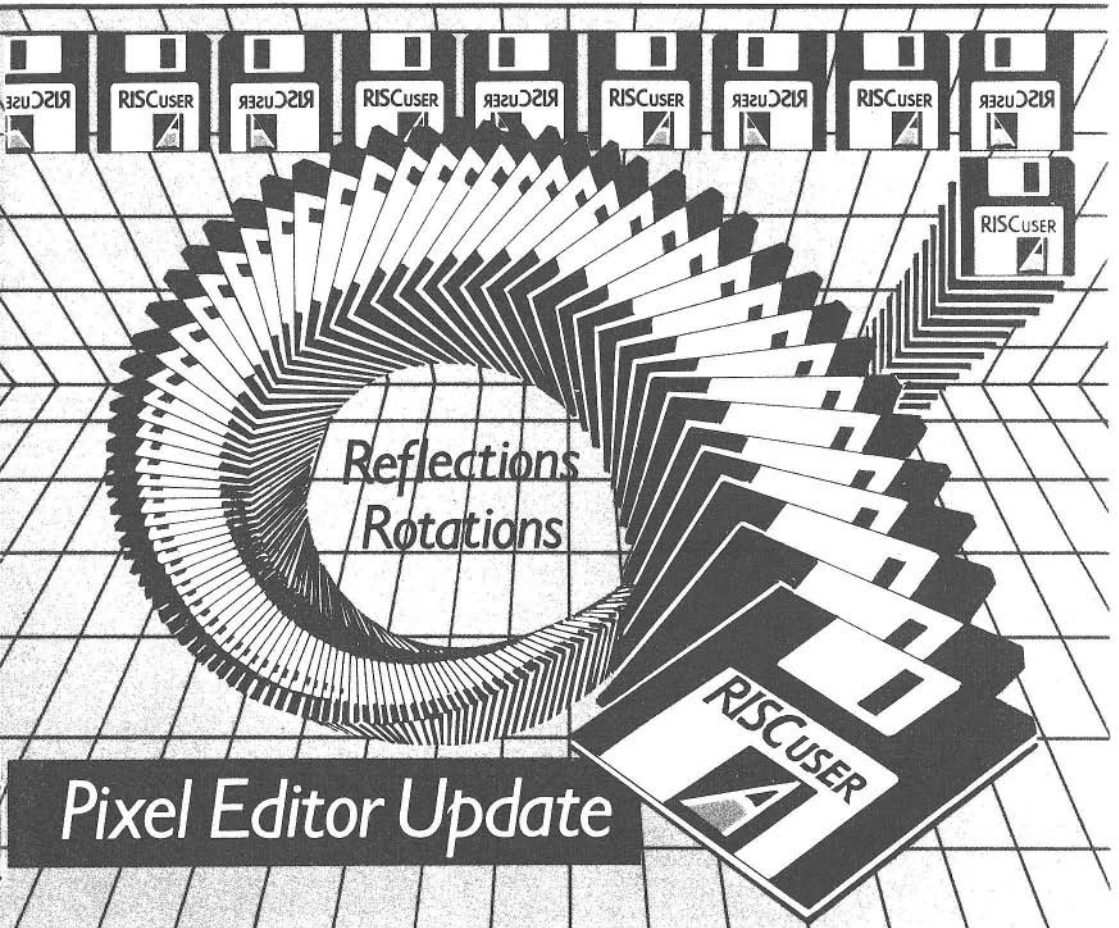
Volume 1
Issue 8

July/
August
1988

Price £1.20

# RISC USER




## Pixel Editor Update

THE MAGAZINE AND SUPPORT GROUP
EXCLUSIVELY FOR USERS OF THE ARCHIMEDES

Volume 1 Issue 8
July/August 1988

# RISC USER

## CONTENTS



**RISC User is published by BEEBUG Ltd.**
**Co-Editors: Mike Williams, Dr Lee Calcraft**
**Assistant Editor: Kristina Lucas**
**Technical Editor: David Spencer**
**Production Assistant: Yolanda Turuelo**
**Technical Assistant: Lance Allison**
**Subscriptions: Mandy Mileham**

BEEBUG, Dolphin Place, Holywell Hill, St.Albans, Herts
AL1 1EX. Tel. St.Albans (0727) 40303



Printed by Arlon House

The Archimedes Magazine
and Support Group.

# EDITORIAL

As Acorn know full well, in the eyes of the general public, a computer is as good, or as bad, as the software which it runs. And their efforts to provide the Archimedes with 6502 and PC emulators does great credit to them, since it vastly increases the number of packages which will run on the Arc. The idea of providing Arc purchasers with a free word processor was also creditable. The implementation, however, left a great deal to be desired, and Acorn has now abandoned ArcWriter. New purchasers of the Archimedes will now receive a 50% discount offer on 1st Word Plus - a much more worthy product.

If you really want to see what the Archimedes can do, you may be interested in two discs retailing at just £7.50 for the pair. These Demo Discs from Clares show just what can be done with the Archimedes, both visually and acoustically. The discs contain a number of graphics demonstrations accompanied by sampled sounds. One disc contains two animations - a full-feature Newton's Cradle (complete with cradle and clicking sound), and their "Crystal Gallery". Like the Cradle demo this is also ray-traced, and features a number of rotating green crystal blocks. Amongst a number of demos on the second disc is an excellent animation of a 3D mask. This swoops around the screen at considerable speed and then bursts into thousands of pieces which hover before settling to the ground. The astounding thing about the Mask sequence is that it is all performed in real time - such is the power of the ARM!

This issue of RISC User is a two month issue covering both July and August
(RISC User is published 10 times a year).
The next issue will be that for September.

# NewsNewsNewsNews NewsNew

## DTP ON ITS WAY

GST, the company that produced 1st Word Plus for Acorn, is working on a complete Desk Top Publishing (DTP) package for the Archimedes. The new system is based on the *Timeworks* DTP package for the IBM PC, and because of its advanced nature will only run under the new Arthur 2 operating system. GST claim the system will be on display at the PCW show in September, although it won't be ready for sale until the end of the year. RISC User will bring you more details as soon as they are available.

## OFF TO A FLYING START

The database management package, *Flying Start II*, has just been released in an Archimedes specific form. The software was previously available for the IBM PC, and could be run using the MS-DOS emulator on the Archimedes (see RISC User Issue 5). However, the new version runs on the native Archimedes, and can therefore make full use of the power and speed of the computer. Mitre Software Ltd., who produce *Flying Start II*, claim that the package can be used by anybody, and that the possible applications range from accounting to performance analysis. *Flying* Start II *costs £69.95 inc. VAT, and a* special 'trial version' can be purchased by would-be buyers. The package is available either through Acorn dealers, including BEEBUG, or direct from Mitre Software Ltd., International House, 26 Creechurch Lane, London EC3A 5BA, phone 01-283 4646.

## PLAY ALONG WITH MINERVA

Minerva Systems have taken a break from business related software to produce two games for the Archimedes. The first of these, *Hoverbod*, is a two dimensional maze adventure along the lines of Manic Miner. The fun of *Hoverbod* lies in the excellent graphics, with cute creatures called Ibbles and Squibbles chasing you round the planet Zingle.

The second game is a version of the arcade classic *Missile Control*, in which you must save your home planet from destruction by shooting down incoming missiles. The games cost £14.95 each, and you can get them either from your local dealer, or direct from Minerva Systems, 69 Sidwell Street, Exeter EX4 6PH, phone (0392) 37756.

## ARCHIMEDES LOGO

*Archimedes Logo* is now available from Logotron, the company that produced Logo for the Master Compact. The software has been totally rewritten compared with the BBC version, and makes full use of the advanced graphics and sound of the Archimedes. Nevertheless, any extension modules written for the older BBC logo can still be used with the new version. *Archimedes Logo* comes on a 3.5" disc with both a tutorial and a reference manual, the whole system costing £71.88 inc. VAT. A network version including site licence is available at £345 inc. VAT. Logotron are at Dales Brewery, Gwydir Street, Cambridge CB1 2LJ, phone (0223) 323656.

## GRAPHICS DEMONSTRATION

Clares Micro Supplies has issued two discs that contain demonstrations of the sound and graphics capabilities of the Archimedes. The discs, which cost £7.50 for the pair, include such things as digitised speech, real time 3D graphics, a Newton's cradle, and a 'flight through a wire frame city'. Clares are at 98 Middlewich Road, Northwich, Cheshire CW9 7DA, phone (0606) 48511.

## LATEST NEWS

Clares Micro Supplies is working on a new version of Artisan to be known as Artisan Professional. This is not intended as a replacement for Artisan, but to offer enhanced drawing facilities in the 256 colour modes for the more demanding user. Artisan Professional is also expected to include a high quality printer dump for the Integrex 132 colour printer (reviewed in this issue.). No announcement on availability has yet been made.

Another novel development for the Archimedes is on its way from Computer Concepts. This is an add-on which will turn your computer system into a sophisticated FAX machine. A fully working version of this device is expected to be on display at the PCW show in September. RU

# PipeDream

PipeDream is now available on the Acorn Archimedes. It provides comprehensive word processing, spreadsheet and database facilities integrated in a way only dreamed of before.

With other integrated packages, you have to divide your work into artificial sections, such as text, numbers and calculation, and database.

With PipeDream, you compose your work in the order you want to print it, with text and numbers all together in one document. Incorporate calculations directly into paragraphs of text and formatted paragraphs directly into spreadsheets.

PipeDream is ideal for all tasks involving words and numbers. From writing thank-you letters to encyclopaediae, invoices to cash flow forecasts, stamp-collection records to inventory management, film scripts to mail-shots.

PipeDream is 100% file and keystroke compatible with Z88 PipeDream and PipeDream on the IBM PC. It is also compatible with View Professional on the BBC microcomputer. You can create documents on any of these computers, transfer the files to any other and continue working, with no loss of information. No other software enables you to share your files with all these computers.

PipeDream for the Acorn Archimedes costs £99 + VAT.

It is not possible to detail all of PipeDream's facilities here. For full details or to order PipeDream call us on 0954 211472 or write to us at Colton Software, Broadway House, 149-151 St Neots Road, Cambridge CB3 7QJ.

PipeDream - power at your fingertips.

# IMPULSE REVEALED

**The latest on Computer Concepts' new operating system for the Archimedes by our roving reporter, Arthur A.Noyed.**

As you will probably be aware, Computer Concepts has announced work on an entirely new operating system for the Archimedes. It is claimed that the Arc's Arthur 1.2 OS is not up to the requirements imposed by their applications software currently under development. This includes a sophisticated word processor and desktop publishing package designed for use with a laser printer.

There are a number of areas in which Arthur 1.2 falls short of CC's requirements, and in particular the provision of full multi-tasking, the associated use of multi-processor systems, and the implementation of PostScript primitives and extensions (PostScript is a special page description language used for driving laser printers). All these features, and more besides, are planned to be implemented in the proposed new operating system, named **Impulse.** And although it is believed that Arthur series two, now under development at Acorn, will go some way to meeting these requirements itself, it is not due for release until early next year.

One of the major features of Impulse is true multi-tasking. Arthur two is believed to permit multi-tasking, but only through the WIMP manager, and although enforcing the use of WIMPs in this way will result in a greater uniformity in the Archimedes world, it does not permit multi-tasking in the widest sense. With Impulse on the other hand, it will be possible to engage in a number of different tasks, each of which might be a separate (and co-resident) Basic program, in a manner not dependent on repeated polling of the WIMP.

There are of course many uses for such a facility. Simple examples include background tasks such as printer spooling, disc cacheing, or any program that is computationally intensive over long periods of time, and where it is disadvantageous to have the computer tied up while it does the calculations. Multi-tasking has other less obvious benefits. For example, consider the case of the database program; one task may be searching the database for some required numeric data, another might be sorting a section of the database, while a third might be accepting user input and displaying a set of records. All of this could be achieved with relative ease using Impulse.

Impulse achieves multi-tasking by splitting each task into a number of so-called task *threads.* Each thread in a given task has the entire address space of that task at its disposal, but may not directly access the memory allocated to other tasks - all this is handled by the Arc's MEMC chip, which was specifically designed for such work. Under Impulse, memory resources are isolated to the extent that even if one task crashes, other tasks will continue.

Not only is Impulse multi-tasking, but it is also a distributed multi-processor operating system. This means that task threads can be assigned not only to time slices of the first processor, but to any attached processor, and indeed to any other computer attached to the system. Impulse will even organise task splitting between a number of Archimedes machines joined by no more than a serial link - at least that is the theory. In practice, at the time of writing, Impulse is not yet complete, but we understand that there will be Impulse-based systems on display at September's PCW show, including a Fax card for the Arc (which will also work under Arthur 1.2), and a RISC processor card for PC compatibles.

We believe that the main outlet for Impulse will be as part of CC's various application software now under development. But CC are also willing to licence the use of Impulse at a low cost to software houses wishing to market products which make use of it.

*Further details may be obtained from Computer Concepts on (0442) 63933.*

RU

# ANIMATING ARCHIE (Part 1)

by Lee Calcraft

**Our new short series on Archimedes animation techniques begins with a look at the use of sprites.**

The Archimedes is a first class machine for those interested in animation. It provides an extensive colour palette and offers the user a variety of high resolution graphics modes. But above all else it is *fast*, and speed is the byword where animation is concerned. In order to give a smooth impression of movement, the presented image must change at a rate of many times per second. For the sake of comparison, television pictures are refreshed 25 times a second (each picture consisting of a pair of interlaced frames).

Routines written in ARM code are easily capable of displaying high resolution colour pictures at comparable frame rates. But we do not need to resort to ARM code for all animation applications. The high speed of Basic V used in conjunction with Archimedes sprites is more than sufficient for many applications, as I hope to show in due course. Also, later in the series we will look at other ways of creating animated sequences using techniques similar to those used in Acorn's Animated Molecule demonstration (distributed on Risc User discs 5 and 6). But for the moment, we begin with a brief introduction to sprites.

## CREATING SPRITES

Sprites are multi-coloured graphics objects which may be manipulated in a variety of ways directly from Basic V or from machine code. On the Archimedes, sprites are stored in a special area of RAM configured by the user with:

```
*Configure SpriteSize
```

where n is the number of *pages* of RAM reserved (one page is 8K on a 300 series machine, or 32K on a 400 series). Once RAM has been allocated, you can go ahead and create a set of sprites. This can be done in one of two ways. You can either use a sprite editor, such as that provided on the Welcome disc (see the *User Guide*, page 165), or the RISC User Pixel Editor (RISC User Issue 4 - with enhancements of particular relevance to this series, described in the present issue), or you can "grab" a part of the screen, and use this as a sprite.

For the moment we will use the latter method. Our sprite will be a blue shaded sphere, generated using the procedure PROCsphere from this month's Visuals. Listing 1 contains a complete program for creating and saving this sprite. Note in particular the two MOVE statements in line 90. These serve to define an area of screen which the command *SGET (line 100) grabs as a sprite. When the program is run, a blue shaded sphere will be drawn at the centre of the screen, and saved as a sprite named BlueBall in a file called SBlueBall.

**Listing 1**

```
  10 REM >1-1Anim2
  20 REM Creates Sphere Sprite
  30 :
  40 MODE15
  50 *SNEW
  60 Z%=0+8+32:R%=50
  70 ORIGIN 640,512
  80 PROCcirc(R%,30,30,2)
  90 MOVE -R%,-R%:MOVE R%,R%
 100 *SGET BlueBall
 110 *SSAVE SBlueBall
 120 END
 130 :
 140 DEFPROCcirc(R%,L1%,L2%,S%)
 150 FOR Y%=R% TO -R% STEP -4
 160 A%=(SQR(R%*R%-Y%*Y%)DIV S%)*S%
 170 FOR X%=-A% TO A% STEP S%
 180 P1%=DEG ASN(X%/R%)
 190 P2%=DEG ASN(Y%/R%)
 200 D1=ABS(P1%-L1%):D2=ABS(P2%-L2%)
 210 C%=7.99-SQR(D1*D1+D2*D2)/14-RND(1)
 220 IFC%<0 THENC%=0
 230 GCOL0,Z%+(C% AND4)*5.25 TINT(C% AN
D 3)*64
 240 PLOT69,X%,Y%:NEXT:NEXT
 250 ENDPROC
```

To check for the presence of this sprite in the sprite area, we can use the command:

```
*SLIST
```

This just catalogues the sprite area. For hard technical information there is the additional command:

```
*SINFO
```

This gives the amount of RAM allocated to sprite use, the quantity remaining free, and the number of currently defined sprites.

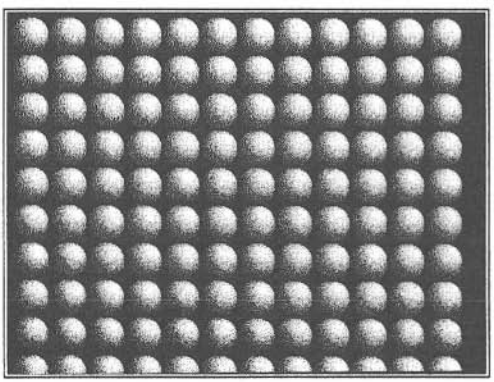
To save the whole set of sprites currently in the sprite area at any time, you can use the command:

```
*SSAVE <filename>
```

as we have done in line 110 of listing 1. Similarly, the command:

```
*SLOAD <filename>
```

will load in a set of sprites. Additionally, the commands ***SNEW** and ***SDELETE <name>** will clear the whole sprite area, or delete a single named sprite, respectively. In possession of this vital information, we can now move on to sprite plotting.



## PLOTTING SPRITES

From the animation point of view, the most important sprite commands are those used to place sprites on the screen. This is achieved with a variant of the PLOT command. To place a sprite at a given position on the screen, you must first ensure that it is present in the sprite area, and then issue the command:

```
*SCHOOSE <name>
```

This makes the named sprite the currently selected sprite, and all further plotting instructions will now refer to it.

The syntax for sprite plotting is:

```
PLOT n,x,y
```

where n is the PLOT number in the range 232 to 239, and x and y are the plot co-ordinates (absolute or relative) of the bottom lefthand corner of the sprite. The most important of the eight possible calls for the moment are PLOT 233 and PLOT 237 (&E9 and &ED in hex). These both place a sprite on the screen; the first using relative graphics co-ordinates, the second, absolute.

**Listing 2**

```
 10 REM >1-2Anim2
 20 REM Plots 120 Blue Spheres
 30 REM Using BlueBall Sprite
 40 :
 50 MODE15
 60 *SLOAD SblueBall
 70 *SCHOOSE BlueBall
 80 FOR X%=0 TO 1100 STEP 100
 90 FOR Y%=0 TO 900 STEP 100
100 PLOT &ED,X%,Y%
110 NEXT:NEXT
```

To plot our BlueBall sprite at position 200,500, we could use the following:

```
*SCHOOSE BlueBall
PLOT &ED,200,500
```

To give you an idea of how fast the sprite plotting routine is, try the program in listing 2. This places 120 blue spheres on the screen in less than half a second.

## MOVING THE SPHERE

If we wish to animate the image in the simplest sense of making the ball appear to move around the screen, then we must repeatedly display and delete the image in a succession of positions along the path chosen for the ball. The best way to delete the sprite is to plot it in so-called Exclusive OR mode (set up with GCOL 3,n where n may be any colour

number), and then overwrite it with its own image in such a way that it cancels itself out. Alternatively we could overwrite the sprite with a rectangle of the background colour. But this is not so flexible as the former method.

Listing 3 gives a short routine to repeatedly move the blue sphere across the screen. Movement is exceptionally smooth, and the speed of traverse can be altered by adjusting the STEP size in line 110. There are a number of points to note about the program. Firstly it plots the sprite twice at the same position in lines 120 and 140 with every cycle of the loop. The first PLOT places the sprite on the screen, and the second removes it. In between the two PLOTs at line 130 is a WAIT statement. This ensures that sprite drawing is linked to the VDU's frame scan rate. It also provides a useful way of ensuring that the sprite is on the screen for a finite period of time before it is wiped off again. The precise effect of the WAIT statement is to cause the program to pause until the end of the current frame scan.

**Listing 3**

```
 10 REM >1-3Anim2
 20 REM Moves Sphere Across Screen
 30 REM Using BlueBall Sprite
 40 :
 50 MODE15
 60 *SLOAD SblueBall
 70 *SCHOOSE BlueBall
 80 Y%=500
 90 GCOL 3,0
100 REPEAT
110 FOR X%=0 TO 1279 STEP 8
120 PLOT &ED,X%,Y%
130 WAIT
140 PLOT &ED,X%,Y%
150 NEXT
160 UNTIL FALSE
```

A second point worth noting is that as the sphere reaches the far right of the screen it appears to continue to move, and goes "behind" the screen border. This so-called "clipping" is performed automatically, and will even work correctly within user-defined graphics windows. To test this you could insert the following line into program 3:

```
55 VDU24,200;200;800;800;
```

As you can see, this window behaves exactly like a physical window, letting you see the sphere as it drifts past, and hiding it when it is to the left or right of the window.

## INCREASING SOPHISTICATION

So far we have only made our sprite move linearly across the screen. By varying the plotting loops used with the program in listing 3 it is possible to send the sprite on any desired trajectory. You could send it round in circles or make it appear to bounce around the screen. For example, if you add the line:

```
115 Y%=400+400*SIN(X%/200)
```

the trajectory will be sinusoidal. Even so this is still a very primitive form of animation. Indeed one might argue that it is not animation at all (is a stone animated when it is thrown?). To put life into an object it must appear to change in aspect. To achieve this using sprites we must define a number of sprites, which when displayed in sequence will simulate such a change.

We might for example create a number of sprites of a face with a changing expression. We could then display these in sequence to create the desired effect. All this is quite feasible on the Archimedes, though we cannot use ***SCHOOSE <name>** to select each sprite in sequence; the call is not fast enough. But fortunately there is an alternative command which will do the job quite adequately:

```
VDU23,0,n|
```

This will select the sprite whose name is the number n, where n may be any integer between 0 and 255. It is then a simple matter to put this instruction into a loop which sequences through the required range of values of n. The trick of course is to create the sprites automatically beforehand wherever possible, otherwise the task can be quite time consuming. But more of this next month. RU

# Basic V — THE ERROR OF ITS WAYS

**This month in the Basic V spot Mike Williams investigates the enhanced error handling available on the Arc.**

Before the advent of the Archimedes and Basic V, error trapping in BBC Basic was of very limited value. Most programs written in BBC Basic make extensive use of procedures and functions together with FOR-NEXT and REPEAT-UNTIL loops. And yet whenever any error occurs, Basic's stack is effectively cleared, and hence all knowledge of the program's position within a loop, procedure or function is lost. Thus most programmers relegate the use of ON ERROR to detecting syntax errors during development, and the use of Escape to terminate execution of a program.

Now this is a shame, as Basic is able to trap many other errors (divide by zero for example) which could be of considerable help if only the limitations described above could be overcome. Instead, programmers have had to write their own error trapping routines (for example, checking for a zero divisor before proceeding with a division).

Basic V changes all that and implements error trapping as it should have been in the first place. The stack is no longer wiped clear when an error occurs, and it is now quite easy to implement a hierarchical error trapping system.

At any time during the execution of a program, the current error status (a form of pointer to the currently active error trapping routine) may be saved to the stack and a new error trapping routine established. A further instruction may subsequently be used to restore the original error status by retrieving this from the stack. Using this simple concept, error trapping may be nested to whatever depth is required (within the capacity of the stack).

To see how this works, consider a short (rather artificial) program to divide two numbers and print the result.

```
100 MODE12: ON ERROR PROCglobal: END
110 LOCAL ERROR
120 ON ERROR LOCAL PROCerror: END
130 INPUT'"Enter two numbers: " X,Y
140 Z=X/Y
150 PRINT X,Y,Z
160 RESTORE ERROR
170 END
180:
190 DEF PROCglobal
200 REPORT:PRINT" at line ";ERL
210 END
220:
230 DEF PROCerror
240 IF ERR=18 PRINT"Zero divisor"
250 ENDPROC
```

Line 100 implements global error trapping as usual. The LOCAL ERROR statement at line 110 saves the current *error status* (i.e. the error trapping set up in line 100), and line 120 creates a local error trapping response (ON ERROR LOCAL is just like ON ERROR as far as syntax is concerned). The program then prompts for two numbers, calculates their quotient and prints both numbers and the result. At the end of this routine, the RESTORE ERROR statement retrieves the previous error status from the stack thus re-establishing global error trapping.

Should the second number be zero, Basic will detect a *Divide by zero* error and invoke the currently active error trapping routine (line 120). This calls a procedure to display a suitable message and then exit.

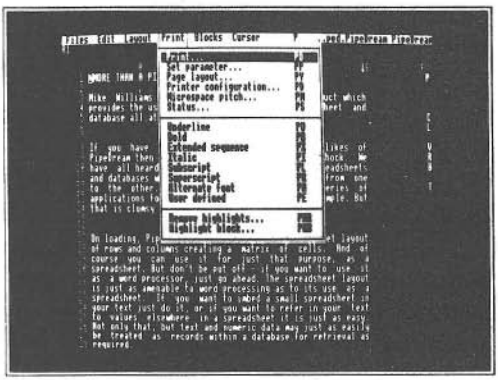
Although the local error trap may have been created for a specific purpose, remember that all errors regardless will now be directed to this routine. In our example all other errors terminate the program forthwith (Escape for example).

Omitting the END following PROCerror in line 120 has the interesting effect of automatically and usefully repeating the routine in the event of division by zero, but unfortunately the same action will also follow the occurrence of any other error (such as Escape) at this stage. Nevertheless, this can be a useful idea as we shall see when we conclude our discussion of error trapping next time. RU

# MORE THAN A PIPEDREAM

**Mike Williams takes a look at a unique product which provides the user with a word processor, spreadsheet and database all at the same time.**

If you have not previously encountered the likes of PipeDream then you may be in for something of a shock. We have all heard of integrated word processors, spreadsheets and databases which allow data to be transferred from one to the other. Computer Concepts' Inter-ROM series of applications for the BBC micro is just such an example. But that is clumsy compared with PipeDream.




On loading, PipeDream presents a typical spreadsheet layout of rows and columns creating a matrix of cells. And of course you can use it for just that purpose, as a spreadsheet. But don't be put off - if you want to use it as a word processor, just go ahead. The spreadsheet layout is just as amenable to word processing as to spreadsheet use. If you want to embed a small spreadsheet in your text just do it, or if you want to refer in your text to values elsewhere in a spreadsheet it is just as easy. Not only that, but text and numeric data may just as easily be treated as records within a database for retrieval as required.

## HISTORY

PipeDream is an evolutionary product. Its origins lie in View, Acorn's word processor for the BBC micro and Master series, and it is written by the same author, Mark Colton. PipeDream as a concept first appeared as View Professional for the BBC micro, and subsequently was adopted and bundled in with the Cambridge Computer Z88 where it has won high praise. More recently PipeDream has been released for the IBM PC and compatibles (that version will run on the Archimedes under the PC emulator), and now a dedicated ARM version is available.

PipeDream for the Arc consists of a disc and 325 page manual supplied in a less than sturdy cardboard box complete with dinky metal hinges. The production of the manual is to a high standard, and the whole appearance, apart from the collapsed box, is that of a quality product.

## USING PIPEDREAM

It is best to think of PipeDream as a sheet on which text, spreadsheets and databases may all be set up. A menu bar at the head of the screen gives access to to six pull-down menus. Any of these may be accessed by pressing the Alt key and the initial letter of the menu name. A scrolling bar in each menu is then used to select any option, and a good many of these generate a further dialogue box.

But that's not all. All the menu options can be selected by direct keyboard input, and many may also be selected using the function keys. Indeed, users of View will find strong echoes of that software in PipeDream's function key layout.

## WORKING WITH TEXT

The first step in using PipeDream is learning how to move about the sheet, up, down left and right. Text may be entered starting in any column and will extend to the right till the right-hand margin for that column is reached. Multiple column format is easily achieved with separate right-hand margins for each column.

All the expected word processing operations are catered for and work well. Blocks of text are highlighted by moving to the start and the end, and may then be moved, copied or deleted. Text can also be surrounded by special markers to indicate the use of bold, italics etc on printing.

You cannot set tab stops and use Tab in the accepted way, as the Tab key is used to move

over the underlying grid. The columns can be used as a substitute, but indenting the start of paragraphs is impossible other than by using spaces. On the plus side, PipeDream has the facility to recover accidentally deleted text using the Paste command, and the same facility may be used deliberately to provide a cut and paste function.

Apart from tabs, I find that using PipeDream as a word processor offers much the same facilities as my favourite View. But altering column widths to set up a table is not as convenient as editing a ruler, and the way PipeDream handles word wrap in relation to columns takes some getting used to. I also feel a built-in spelling checker is almost essential for any self-respecting word processor these days, if not a thesaurus too. However, the former at least is said to be on the way.

## WORKING WITH NUMBERS

Numbers and expressions, including a wide range of special functions, may be entered into any slot if first designated as an expression slot (this can also be set up as a default). Expressions may consist of numbers, dates, functions, conditions, strings, ranges, lists and slot references (a range is a consecutive set of cells, a list of individual cells). These may be entered individually or combined using various mathematical and conditional operators. Fourteen mathematical functions are provided, plus 11 financial functions (e.g. loan repayment, cashflow etc.), 13 general functions and 9 database functions.

I set up a spreadsheet application duplicating one I use regularly with another package. Entering expressions of various kinds was reasonably straightforward, as was the replication of formulae with correct adjustment of cell references. These references can be entered into expressions by moving the cursor to the appropriate cell, but this is not as automatic as in Logistix for example, and requires an extra key depression (Ctrl-f6).

Recalculation is automatic whenever an expression is modified, and quite fast enough in the application I tried. Single cells and blocks of cells may be formatted as required (number of decimal places for example), and a useful feature enables leading or trailing characters (such as £ and % signs) to be added to blocks of cells automatically.

I have used Logistix (see review in RISC User Issue 3) quite extensively on the Archimedes, and in my view PipeDream is neither quite as convenient to use for spreadsheet work nor as powerful. But then Logistix does not have PipeDream's integration of word processor and spreadsheet facilities.

## DATABASE FUNCTIONS

Any data may be inserted in rows and columns to form a database. Values may be sorted or matched, and some nine special functions will perform tasks such as totalling and locating maximum and minimum values. The facilities are a long way short of any full-blown database system, but useful none the less in the context.

## OTHER FEATURES

The menu system works well and provides a wealth of functions. Page layouts are readily created with headers and footers as required with good control over printing including various highlights such as italic and underlined text. Printer drivers are supplied for the Juki 6100 and Epson FX80, but other drivers are readily created by editing the files supplied using PipeDream itself.There are also facilities to enable files to be transferred between PipeDream and Lotus 1-2-3, and of course there is complete compatibility between the Archimedes version of PipeDream and those running on the PC or Z88.

## CONCLUSIONS

PipeDream has much to commend it, not least an excellent manual. There may be word processors or spreadsheets which offer more in their respective applications, but for its unique combination of talents PipeDream has much going for it.

| | |
|---|---|
| Product | PipeDream |
| Supplier | Colton Software,<br>Broadway House,<br>149-151 St Neots Road,<br>Cambridge CB3 7QJ.<br>Tel. (0954) 211472 |
| Price | £113.85 inc. VAT. |

RU

# RISC USER TOOLBOX (Part 2)

**David Spencer adds a powerful search facility to our Toolbox module.**

In last month's opening article, we gave the basic framework for the RISC User Toolbox module, together with the first utility, a memory editor. This month we are going to add some memory search facilities to our Toolbox. By using these search commands, it is possible to search through any section of memory for a given string, or a sequence of bytes or 32-bit words. When a match has been found, the memory editor is entered at that point. You can then use the editor to make any changes necessary. When finished, you can search for the next occurrence of the target string, or abandon the search.

The listing given this month is not a complete program, but a series of lines to add to last month's listing. Before starting, load in the program from last month and ensure that the line numbering corresponds to that in the article. Before entering the new lines, type:

```
DELETE 2440,2550
```

to remove some lines which are no longer needed. Then enter the listing given here, and save the extended program. Use a different filename to the original, and make sure that you do not accidentally renumber the program. Run the program to reassemble the Toolbox module on disc and load it as described last month.

You will find that there are now three new star commands: *SEARCHB, *SEARCHW, and *SEARCHS to search for bytes, words and strings. The first two commands have similar syntax:

```
*SEARCHB <address> <byte sequence>
*SEARCHW <address> <word sequence>
```

and search memory for the given sequence of bytes (numbers in the range 0-&FF), or 32-bit words respectively. The values are specified in hex (with or without an &), although another base can be used by preceding the number with 'base_' (e.g. 2_10010). Individual numbers should be separated by spaces. For example, to search for the hex bytes &FF, &FE from address &8000, you could use :

```
*SEARCHB 8000 FF FE
```

The search will start from the address given as the first parameter, and will continue until an address that is not mapped into physical RAM is encountered. Each time the specified sequence is found, the editor is entered with the cursor at the start of that sequence. The editor can then be used as if it had been entered using *MEDIT. When you have finished, pressing Escape will abandon the search and quit, while pressing the Copy key will search for the next match in memory.

The third command is of the form:

```
*SEARCHS <address> <search string>
```

and searches for the specified character string in memory. *SEARCHS allows the use of special characters in the search string to provide 'wildcard' matching.

The special characters recognised are:

| | |
|---|---|
| . | to match any character |
| # | to match any digit |
| @ | to match any alphanumeric (A-Z,a-z,0-9 or _) |
| $ | to match a carriage return |
| \|x | to match Ctrl-x (e.g. \|G to match Ctrl-G) |
| \x | to match x, where x is a special character (e.g. '\#' to match '#') |

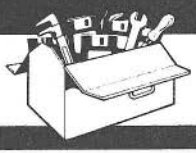
All characters in the search string are significant, including leading and trailing spaces. If there is more than one space between the start address and the search string, then the extra spaces will be taken as part of the search string. The search is not case specific, that is upper and lower case are treated the same, but to force a case sensitive match you can precede any letter with a '\'.

This month's additions also include some changes to the memory editor itself. In particular, you will notice a status line at the bottom of the screen which indicates the editing mode. The editor also starts up more quickly, with only a very short pause between typing *MEDIT and the edit screen appearing. Finally, *MEDIT can now be used without an address, with the value of &8F00 (Basic's default PAGE) being used instead.

***Further Toolbox extensions next month. If you have any suggestions for utilities to include in the RISC User Toolbox, please drop us a line. We will use the best ideas in future issues.***



```
  300 EQUD &10000
  311 EQUS "SearchB":EQUB 0
  312 ALIGN:EQUD searchb:EQUD &FF0002
  314 EQUD sbsyn:EQUD sbhlp
  315 EQUS "SearchW":EQUB 0
  316 ALIGN:EQUD searchw:EQUD &FF0002
  318 EQUD swsyn:EQUD swhlp
  319 EQUS "SearchS":EQUB 0
  320 ALIGN:EQUD searchc:EQUD &FF0002
  322 EQUD scsyn:EQUD schlp
 1370 EQUS "Syntax: Medit [<address>]"
 1381 .sbhlp:EQUS "*SearchB searches for
 a byte sequence.":EQUB 13
 1384 .sbsyn:EQUS "Syntax: SearchB <addr
ess> <byte>...":EQUB 0:ALIGN
 1387 .swhlp:EQUS "*SearchW searches for
 a word sequence.":EQUB 13
 1390 .swsyn:EQUS "Syntax: SearchW <addr
ess> <word>...":EQUB 0:ALIGN
 1393 .schlp:EQUS "*SearchS searches for
 a character sequence.":EQUB 13
 1396 .scsyn:EQUS "Syntax: SearchS <addr
ess> <string>":EQUB 0:ALIGN
 2391 LDR R12,[R12]
 2400 STMFD R13!,{R14}:CMP R1,#0:MOVEQ R
2,#&8F00
 2401 MOVNE R1,R0:BEQ noadd
 2410 MOV R0,#16:SWI "OS_ReadUnsigned"
 2411 .noadd
 2430 BL valadd:SWI &116
 2450 SWI &10C:BL frange:ADR R5,mtxt
 2560 MOV R1,#1:MOV R4,#0
 2620 BNE memedit2:SWI &10A:LDMFD R13!,{
PC}
 2621 .mtxt
 2622 EQUS "Logical memory editor"
 2623 EQUB 0
 2651 EQUS "MSearch":EQUB 0
 2652 EQUS "StrParse":EQUB 0
 2653 EQUS "StrMatch":EQUB 0
 3711 B swi1:B swi2:B swi3
 6460 BL statline:B keydone
 6590 STMFD R13!,{R0-R2}
 6600 LDR R5,[R12,#16]:SWI &111
 6610 SWI &183:SWI &111:SWI &104
 6620 MOV R2,R5:MOV R1,#0
 6630 .st1 LDRB R0,[R2],#1:CMP R0,#32
 6640 BCC st2:SWI "OS_WriteC"
 6650 ADD R1,R1,#1:CMP R1,#70:BNE st1
 6660 BEQ st3
 6670 .st2 SWI &120:ADD R1,R1,#1
 6680 CMP R1,#70:BNE st2
 6690 .st3 LDR R0,[R12]:CMP R0,#0
 6700 ADREQ R0,hextxt:ADRNE R0,asctxt
 6710 SWI "OS_Write0":LDMFD R13!,{R0-R2,
PC}^
 6720 .asctxt EQUS"ASCII ":EQUB17
 6730 EQUB128:EQUB17:EQUB7:EQUB0:ALIGN
 6740 .hextxt EQUS"Hex   ":EQUB17
 6750 EQUB128:EQUB17:EQUB7:EQUB0:ALIGN
 6760 .frange STMFD R13!,{R0-R1,R14}
 6770 BIC R0,R0,#&FF:BIC R0,R0,#&1F00
 6780 MOV R1,R0
 6790 .fr1 SWI "OS_ValidateAddress"
 6800 SUBCC R0,R0,#8192:BCC fr1
 6810 ADD R2,R0,#8192:MOV R0,R1
 6820 .fr2 ADD R1,R1,#8192
 6830 SWI "OS_ValidateAddress"
 6840 BCC fr2:SUB R3,R1,#8192
 6850 LDMFD R13!,{R0-R1,PC}
 6860 .swi1 STMFD R13!,{R1-R9,R14}
 6870 ADD R4,R12,#256:CMP R3,#4
 6880 MOVEQ R5,R2,LSL #2:MOVNE R5,R2
 6890 MOV R2,R2,LSL #2
 6900 SUB R5,R1,R5:.bws1 MOV R6,#0
 6910 MOV R7,#0:.bws2 CMP R3,#4
 6920 LDREQ R8,[R0,R6]:LDREQ R9,[R4,R7]
 6930 LDRNEB R8,[R0,R6]:LDRNEB R9,[R4,R7
]
 6940 ADD R7,R7,#4:ADD R6,R6,R3
 6950 CMP R8,R9:BNE bws3:CMP R7,R2
 6960 BNE bws2:LDMFD R13!,{R1-R9,PC}
 6970 .bws3 ADD R0,R0,R3:CMP R0,R5
 6980 BNE bws1:MOVS R0,#1
 6990 LDMFD R13!,{R1-R9,PC}
 7000 .setxt
 7010 EQUS "Search mode      COPY for ne
xt match, ESCAPE to exit":EQUB 0
 7020 ALIGN
 7030 .searchb MOV R3,#1:B se
 7040 .searchw MOV R3,#4
 7050 .se STMFD R13!,{R14}:LDR R12,[R12]
 7060 MOV R4,#0:STR R4,[R12,#20]
 7070 SUB R4,R1,#1:MOV R1,R0:MOV R0,#16
 7080 ORR R0,R0,#1<<31
 7090 SWI "OS_ReadUnsigned":BL valadd
 7100 STMFD R13!,{R2}:MOV R5,#0
 7110 ADD R6,R12,#256:.se2 CMP R3,#4
 7120 MOVEQ R0,#16:MOVNE R0,#16
 7130 ORRNE R0,R0,#1<<30
 7140 ORR R0,R0,#1<<31
 7150 ADD R1,R1,#1:SWI "OS_ReadUnsigned"
 7160 STR R2,[R6],#4:ADD R5,R5,#1
 7170 CMP R5,R4:BNE se2
 7180 LDMFD R13!,{R0}:MOV R5,R3
 7190 BL frange:MOV R1,R3:MOV R2,R4
 7200 MOV R3,R5
 7210 .se3 BL swi1:BEQ se5
 7220 .se4 LDR R0,[R12,#20]:CMP R0,#0
 7230 BNE se45:SWI "OS_WriteS"
 7240 EQUS "Not Found":EQUW &D0A
 7250 EQUB 0:LDMFD R13!,{PC}
```



```
 7260 .se45 ADD R1,R12,#256:MOV R2,#256
 7270 SWI "OS_BinaryToDecimal":MOV R0,#0
 7280 STRB R0,[R1,R2]:MOV R0,R1:SWI "OS_
Write0"
 7290 SWI "OS_WriteS":EQUS " found"
 7300 EQUW &D0A:EQUB 0:LDMFD R13!,{PC}
 7310 .se5 STMFD R13!,{R0-R3}
 7320 LDR R1,[R12,#20]:ADD R1,R1,#1:STR
R1,[R12,#20]
 7330 MOV R1,#0:.se6 BL frange
 7340 MOV R4,#0:ADR R5,setxt
 7350 SWI &116:SWI &10C:ORR R1,R1,#1
 7360 BL swi0:SWI &11F:SWI &100
 7370 SWI &11F:CMP R2,#27:BNE se7
 7380 SWI &10A:LDMFD R13!,{R0-R3,PC}
 7390 .se7 CMP R2,#&AB:BNE se6
 7400 SWI &10A:LDMFD R13!,{R0-R3}
 7410 ADD R0,R0,R3:B se3
 7420 .valadd
 7430 STMFD R13!,{R0-R2,R14}
 7440 MOV R0,R2:MOV R1,R2
 7450 SWI "OS_ValidateAddress"
 7460 LDMFD R13!,{R0-R2,R14}
 7470 MOVCC PC,R14
 7480 CMP PC,#1<<31:ADR R0,bserr
 7490 LDMFD R13!,{PC}
 7500 .bserr EQUD 0
 7510 EQUS "Bad start address"
 7520 EQUB 0:ALIGN
 7530 .searchc STMFD R13!,{R14}
 7540 LDR R12,[R12]:SUB R4,R1,#1
 7550 MOV R1,#0:STR R1,[R12,#20]
 7560 MOV R1,R0:MOV R0,#16
 7570 ORR R0,R0,#1<<31:SWI "OS_ReadUnsig
ned"
 7580 BL valadd:ADD R1,R1,#1
 7590 STMFD R13!,{R2}:MOV R0,R1
 7600 ADD R1,R12,#256:BL swi2:MOV R4,R2
 7610 LDMFD R13!,{R0}:ADD R1,R12,#256
 7620 BL frange:SUB R2,R3,R4
 7630 .sec1 BL swi3:BEQ sec2
 7640 ADD R0,R0,#1:CMP R0,R2:BNE sec1
 7650 B se4
 7660 .sec2 STMFD R13!,{R0-R2}
 7670 LDR R1,[R12,#20]:ADD R1,R1,#1:STR
R1,[R12,#20]
 7680 MOV R1,#2:.sec3 BL frange
 7690 MOV R4,#0:ADR R5,setxt
 7700 SWI &116:SWI &10C:ORR R1,R1,#1
 7710 BL swi0:SWI &11F:SWI &100
 7720 SWI &11F:CMP R2,#27:BNE sec4
 7730 SWI &10A:LDMFD R13!,{R0-R2,PC}
 7740 .sec4 CMP R2,#&AB:BNE sec3
 7750 SWI &10A:LDMFD R13!,{R0-R2}
 7760 ADD R0,R0,#1:B sec1
 7770 .swi2 STMFD R13!,{R0,R3-R6,R14}
 7780 MOV R3,#0:MOV R4,#1
 7790 .par1 LDRB R5,[R0],#1:CMP R5,#32
 7800 .par15 STRCCB R3,[R1]:MOVCC R2,R3
 7810 LDMCCFD R13!,{R0,R3-R6,PC}
 7820 CMP R5,#ASC".":MOVEQ R5,#&80
 7830 CMP R5,#ASC"#":MOVEQ R5,#&81
 7840 CMP R5,#ASC"@":MOVEQ R5,#&82
 7850 CMP R5,#ASC"$":MOVEQ R5,#&83
 7860 CMP R5,#ASC"\":BNE par2
 7870 LDRB R5,[R0],#1:CMP R5,#32
 7880 BCC par15:ORR R5,R5,#128:B par4
 7890 .par2 CMP R5,#ASC"|":BNE par3
 7900 LDRB R5,[R0],#1:CMP R5,#32
 7910 BCC par15:AND R5,R5,#&DF
 7920 CMP R5,#&40:BCC par1
 7930 CMP R5,#&60:BCS par1
 7940 SUB R5,R5,#&40
 7950 .par3 CMP R5,#ASC"a":BCC par4
 7960 CMP R5,#ASC"z"+1:BCS par4
 7970 AND R5,R5,#&DF
 7980 .par4 ADD R3,R3,#1
 7990 STRB R5,[R1,R4]:ADD R4,R4,#1
 8000 B par1
 8010 .swi3 STMFD R13!,{R0-R6,R14}
 8020 MOV R4,#0:LDRB R5,[R1],#1
 8030 .comp1 CMP R4,R5:BNE comp2
 8040 MOVS R0,#0:LDMFD R13!,{R0-R6,PC}
 8050 .comp2 LDRB R2,[R0],#1:LDRB R3,[R1
],#1
 8060 AND R6,R3,#&80:CMP R3,#&90:BICCS R
3,R3,#&80
 8070 ADD R4,R4,#1:CMP R3,#&80:BCS comp3
 8080 CMP R6,#0:BNE comp21:CMP R2,#ASC"a
"
 8090 BCC comp21:CMP R2,#ASC"z"+1:BCS co
mp21
 8100 AND R2,R2,#&DF
 8110 .comp21 CMP R2,R3:BEQ comp1
 8120 .comp25 MOVS R0,#1:LDMFD R13!,{R0-
R6,PC}
 8130 .comp3 BEQ comp1:CMP R3,#&83
 8140 BNE comp4:CMP R2,#13:BEQ comp1
 8150 B comp25
 8160 .comp4 CMP R2,#&30:BCC comp25
 8170 CMP R3,#&81:BNE comp5
 8180 CMP R2,#&3A:BCC comp1:B comp25
 8190 .comp5 CMP R2,#&3A:BCC comp1
 8200 CMP R2,#&41:BCC comp25
 8210 CMP R2,#&5B:BCC comp1
 8220 CMP R2,#&5F:BEQ comp1
 8230 CMP R2,#&61:BCC comp25
 8240 CMP R2,#&7B:BCC comp1
 8250 B comp25
60000 ]NEXT
60010 SYS "OS_File",10,"TOOLBOX",&FFA,,c
ode,O%
```

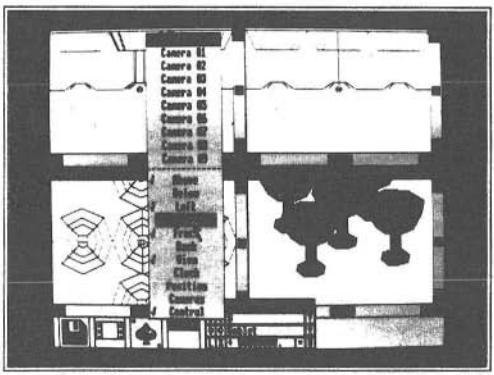
RU

# EUREKA FOR EUCLID

**Mike Williams explores the world of three dimensional graphics through a new software package from Ace Computing.**

Given the Archimedes' immense capacity for colour graphics and its ability to update screen displays rapidly, it is no surprise to find 3D graphics packages beginning to appear. Silicon Vision has already released its 3D Development System (see RISC User Issue 7), but this is just a repackaging of an existing BBC micro program. Newly released Euclid from newcomer Ace Computing would seem to be the first 3D design package written specifically for the Archimedes.

The software is supplied on disc with (currently) a 36 page manual. The heart of Euclid is a relocatable module, and like a good many applications packages nowadays this may be used at two levels: by direct calls to the module as the basis of one's own programs, or via a user-friendly front-end 'Designer' program also supplied on disc. This review will be devoted almost entirely to the latter.



Once selected the Designer presents four windows labelled **Above**, **Front**, **Left** and **View**. The first three represent orthogonal views of the object(s) being created; the View window provides coloured perspective views of the object(s) from a variety of user-selected angles. There are also four large menu icons at the foot of the screen, and a 'Control' panel.

Objects in Euclid are represented hierarchically rather like the ADFS menu structure. The highest level is the complete view. This is then broken down into subsidiary objects, and so on until the basic level of points and faces is reached. The Control panel provides mouse-selected functions to move around an object's structure, and also displays the name of the currently active object.

The control panel also provides for an object to be translated and rotated with reference to the three Cartesian axes (X, Y, and Z), and an object may also be scaled. New objects are created from the bottom up, by creating planes and solids that may be combined together to produce the desired shape. The View window presents views of the object as seen from up to 10 'cameras'. These are positioned and angled by the user who may then switch from camera angle to camera angle at will.

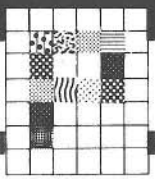
Euclid is very largely mouse controlled, with the menu button selecting a **Functions** menu in any window allowing objects to be created, duplicated, destroyed etc. There are also menus for file operations, camera control and printing (only the Epson FX, MX and RX printers are currently supported), and a so-called **Goodies** menu of miscellaneous functions.

Any system for creating and manipulating 3D objects is bound to take time for the user to achieve a reasonable level of competence and familiarity. That was certainly my experience with Euclid. However, guided by the manual, I found no great difficulty in manipulating one of the example files supplied nor in creating from scratch a representation of a wine glass. The manual did have to be read very carefully in order to achieve the desired results, and could certainly be improved.

The system is well thought out and works well, but does little at present to extend 3D graphics beyond what is already available on the BBC micro (Euclid does not provide texturing or shading for example). The manual also has a tendency to make things seem more difficult than they really are. Euclid certainly represents a good step in the right direction for 3D graphics on the Arc, and at the current inclusive price of £45 is well worth the attention of anyone interested in this field.

| | |
|---|---|
| **Product** | **Euclid** |
| **Supplier** | **Ace Computing**<br>**27 Victoria Road,**<br>**Cambridge CB4 3BW.**<br>**Tel. (0223) 322559** |
| **Price** | **£45 inclusive** |

**RU**



# PIXEL EDITOR UPDATE

**Barry Christie has added a versatile 'cut and paste' facility to his pixel editor published in Issue 4.**

The Full Screen Pixel Editor published in RISC User Issue 4 has proved very popular (this month's RISC User disc contains an excellent example of its use). This update adds a sophisticated set of functions to cut, copy and paste areas of a picture, plus further transformations on the way.

The alterations to the pixel editor are presented here as a series of lines to add to the original program. Before entering these new lines, make sure that you have a copy of the original program loaded into the computer, and that the line numbering is the same as that published in the original article. Then enter the new lines listed here, some of which will overwrite existing lines. When you have finished, save the new program under a different name to the original. This month's disc contains the original pixel editor with the new features added, and also a further enhanced version of the editor with a smarter screen display. The disc also contains some demonstration screens drawn using the original program by RISC User member Lee Thake.

## THE NEW FEATURES

The enhanced pixel editor includes a cut, copy and paste system which you can apply to any part of the screen display. Whenever you are editing a picture, you can 'pick up' any rectangular area of the screen image, and then reproduce it anywhere else within the picture. When an area is selected, it is read into a buffer (in the form of a sprite) and can be used as often as you wish. It is thus possible to draw an image, and then select that image and replicate it many times within the overall picture.

Once an area of the original picture has been selected, it can either be copied, leaving the original intact, or be cut, causing the 'hole' thus created to be filled with the currently selected colour. By using these two options, it is possible to perform both move and copy operations.

The image currently stored in the buffer can be 'pasted' onto the picture at any point. The enhanced pixel editor not only allows an exact copy of the current image to be pasted, but also lets you paste the image reflected in either the X or Y axes, or rotated through any angle about its bottom left-hand corner. This allows quite complex operations to be performed. For example, from a picture of a man in profile, you could copy his body exactly, and then copy his limbs one by one, rotating each one slightly. This would give a second image of the man, with the impression that he had moved slightly from the first image. Such operations prove very useful in animation, where many similar, but subtly different, pictures are needed - this will be covered in our series on animation starting in this issue.

The enhanced editor also lets you save the current image stored in the buffer as a sprite, or load in a previously saved sprite. This is a very useful feature that allows you to build up a library of images that can then be incorporated into other pictures. This also provides a method of saving just part of a picture.

## USING THE ENHANCED EDITOR

All the new features of the pixel editor are controlled by function keys. There are eight function key operations altogether, and these are detailed below. You may find it useful to make a function key strip with the key operations marked on it.

**F1** invokes the copy operation. If F1 is pressed from the edit screen, it will bring up a full-screen version of the picture, with a rectangular box that can be moved around using the mouse. Pressing the *menu* button on the mouse at any time will freeze the bottom left corner of the box, and allow the top right corner to be dragged about, thereby changing the box's horizontal *and* vertical dimensions. As soon as the *menu* button is released, the newly-sized box becomes mobile again. Pressing the *select* button at any time will copy the area enclosed by the box into the image buffer, and remove the box.

**F2** is used for the cut operation. This behaves in exactly the same way as F1, except that when *select* is pressed, the enclosed area is cleared to the currently selected colour after it has been copied into the buffer.

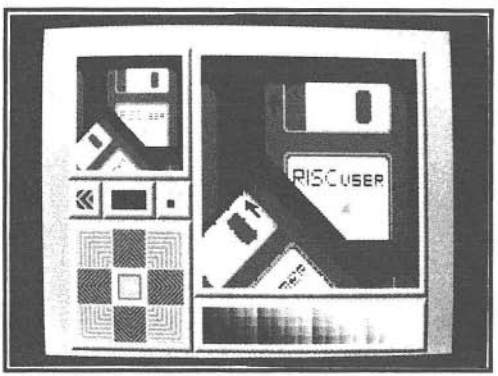
**F5** allows the current buffer contents to be pasted into a picture. On pressing F5, a full

screen picture with box will appear, but this time the box is the shape and size of the stored image. Position the box where you want the image to appear, and press the *select* button. The image will be 'pasted in', overwriting anything already there.

**F6** is similar to F5, except that the image will be reflected in the X axis (turned on its head) before it is pasted in.

**F7** is the same as F6, except that the reflection is in the Y axis.

**F8** allows the image to be rotated through any angle before being pasted in. Pressing F8 brings up the box as before, but this time the *menu* and *adjust* buttons can be used to rotate the box clockwise and anticlockwise respectively. Once the box is positioned and orientated correctly, pressing *select* will paste in the image. Because the rotation routine is written in Basic, it is much slower than the others which are almost instantaneous.




All the above options leave the editor in the full-screen display, so that further cutting and pasting can be done if necessary. To return to the edit screen press the *adjust* button.

**F9** lets you load in a named image. This only works from the options page.

**F11** is the same as F9, but saves the current image using the supplied name.

```
  240 MOUSE mx,my,mb
  260 areachosen=areax>16 AND areax<304
AND areay=1 AND mb<>0:CLS
  351 brushtype=FNbrushkeys
  352 IF brushtype=7 PROCloadbrushes
  353 IF brushtype=8 PROCsavebrushes
  831 brushtype=FNbrushkeys
  832 IF brushtype<7 PROCbrushesfunc
 3330 :
 3340 DEF PROCloadbrushes
 3350 *FX 15,0
 3360 INPUT "Enter <filename> of brush t
o LOAD ... "filename$
 3370 OSCLI("SLOAD "+filename$):PROCcont
inue
 3380 SYS "OS_SpriteOp",13,0,B%,13,1 TO
d,d,d,length
 3390 B%?length=13:OSCLI("SRENAME "+$B%+
" brushsprite")
 3400 SYS "OS_SpriteOp",40,0,"brushsprit
e" TO d,d,d,width,height
 3410 cw%=width*dotsize:ch%=height*4
 3420 ENDPROC
 3430 :
 3440 DEF PROCsavebrushes
 3450 *FX 15,0
 3460 INPUT "Enter <filename> of brush t
o SAVE ... "filename$
 3470 OSCLI("SRENAME brushsprite "+filen
ame$)
 3480 OSCLI("SSAVE "+filename$):PROCcont
inue
 3490 OSCLI("SRENAME "+filename$+" brush
sprite")
 3500 ENDPROC
 3510 :
 3520 DEF PROCcheckmouse
 3530 MOUSE mx,my,mb
 3540 ENDPROC
 3550 :
 3560 DEF PROCbrushesfunc
 3570 PROCdriversdisplay(2,2)
 3580 MOUSE RECTANGLE -1280,-1024,2560,2
048
 3590 rotate=FALSE
 3600 WHILE brushtype<>9
 3610 cx%=mx
 3620 cy%=my
 3630 ca%=0
 3640 CASE brushtype OF
 3650 WHEN 1,2 : PROCbrushpick2
 3660 WHEN 3,6 : PROCbrushplot("")
 3670 WHEN 4   : PROCbrushplot("SFLIPX b
rushsprite")
 3680 WHEN 5   : PROCbrushplot("SFLIPY b
rushsprite")
 3690 ENDCASE
 3700 brushtype=FNbrushkeys
 3710 ENDWHILE
```

```
 3720 MOUSE RECTANGLE 0,0,1280,1024
 3730 IF rotate THEN PROCeditorscreen
 3740 PROCdisplayfunc(0,0)
 3750 ENDPROC
 3760 :
 3770 DEF PROCbrushpick2
 3780 cw%=320:ch%=256:PROCbrushpick(TRUE
)
 3790 PLOT 4,cx%,cy%:GCOL 0,gcol TINT ti
nt
 3800 PLOT 0,cw%,ch%:VDU 7:*SGET brushsp
rite
 3810 IF brushtype=2 THEN RECTANGLE FILL
 cx%,cy%,cw%,ch%
 3820 ENDPROC
 3830 :
 3840 DEF PROCbrushplot(flip$)
 3850 OSCLI(flip$)
 3860 PROCbrushpick(FALSE)
 3870 IF brushtype=6 THEN
 3880 PROCbrushrotate
 3890 ELSE
 3900 GCOL 0:*SCHOOSE brushsprite
 3910 PLOT &ED,mx,my
 3920 ENDIF
 3930 OSCLI(flip$)
 3940 ENDPROC
 3950 :
 3960 DEF PROCbrushrotate
 3970 cos=COS(ca):sin=SIN(ca)
 3980 *FX 112,1
 3990 GCOL 0:*SCHOOSE brushsprite
 4000 PLOT &ED,0,0
 4010 FOR X%=0 TO cw% STEP 2
 4020 FOR Y%=0 TO ch% STEP 4
 4030 *FX 112,1
 4040 GCOL POINT(X%,Y%) TINT (TINT(X%,Y%
))
 4050 *FX 112,2
 4060 RECTANGLE cx%+X%*cos-Y%*sin,cy%+X%
*sin+Y%*cos,4,4
 4070 NEXT Y%
 4080 NEXT X%
 4090 rotate=TRUE:VDU 7
 4100 ENDPROC
 4110 :
 4120 DEF PROCbrushpick(choose)
 4130 PROCbrusharea
 4140 REPEAT
 4150 PROCcheckmouse
 4160 IF mb=2 AND choose THEN PROCbrushs
ize ELSE PROCbrushmove
 4170 UNTIL mb=4
 4180 PROCbrusharea
 4190 ENDPROC
 4200 :
 4210 DEF PROCbrusharea
 4220 GCOL 3,63
 4230 PLOT 4,cx%,cy%
 4240 PROCbrushrotatepoint(cw%,  0)
 4250 PROCbrushrotatepoint(cw%,ch%)
 4260 PROCbrushrotatepoint(  0,ch%)
 4270 PLOT 5,cx%,cy%
 4280 ENDPROC
 4290 :
 4300 DEF PROCbrushrotatepoint(rx%,ry%)
 4310 ca=RAD(ca%)
 4320 PLOT 5,cx%+rx%*COS(ca)-ry%*SIN(ca)
,cy%+rx%*SIN(ca)+ry%*COS(ca)
 4330 ENDPROC
 4340 :
 4350 DEF PROCbrushmove
 4360 IF cx%<>mx OR cy%<>my THEN
 4370 PROCbrusharea:cx%=mx:cy%=my:PROCbr
usharea
 4380 ENDIF
 4390 IF brushtype=6 AND (mb=1 OR mb=2)
THEN
 4400 PROCbrusharea:ca%+=3-2*mb:PROCbrus
harea
 4410 ENDIF
 4420 ENDPROC
 4430 :
 4440 DEF PROCbrushsize
 4450 MOUSE TO cx%,cy%:mx=cx%:my=cy%
 4460 REPEAT
 4470 IF mx<>cx% OR my<>cy% THEN
 4480 PROCbrusharea:cw%+=mx-cx%:ch%+=my-
cy%
 4490 PROCbrusharea
 4500 ENDIF
 4510 PROCcheckmouse:MOUSE TO cx%,cy%
 4520 UNTIL mb<>2
 4530 ENDPROC
 4540 :
 4550 DEF FNbrushkeys
 4560 PROCcheckmouse
 4570 CASE TRUE OF
 4580 WHEN INKEY(-114) : =1
 4590 WHEN INKEY(-115) : =2
 4600 WHEN INKEY(-117) : =3
 4610 WHEN INKEY(-118) : =4
 4620 WHEN INKEY(- 23) : =5
 4630 WHEN INKEY(-119) : =6
 4640 WHEN INKEY(-120) : =7
 4650 WHEN INKEY(- 29) : =8
 4660 WHEN mb=1  : =9
 4670 ENDCASE
 4680 =10
```

RU

# USING THE PC EMULATOR

**Ian Whiting begins a new series of occasional articles on the PC Emulator.**

The Acorn Archimedes PC Emulator is a software simulation of the popular IBM PC, and was reviewed in RISC User Issue 2. Since then Acorn has released a number of updates which claim an improved performance and implementation. These short articles will cover the use of the Emulator and various aspects of MS-DOS from an Archimedes point of view.

## CONFIGURING FOR THE EMULATOR

The standard IBM PC can have up to 640K of main memory. With the Emulator we can only get 640K of user RAM when using an A440 with 4Mb of main memory.

A 1Mb Archimedes will simulate an IBM PC with up to 550K of main memory. To achieve 550K you must carefully set up the Archimedes before loading the Emulator. On power up, exit from the Desktop (if necessary) and list all existing modules using *MODULES, and the configuration using *STATUS. Remove as many modules as possible using *UNPLUG <name> (e.g. *UNPLUG DESKTOP) - on my A310 I unplug all modules except the UtilityModule, FileSwitch, Basic, ADFS and SoundChannels (apparently still needed though sound is not supported). The latest version of the PC Emulator reconfigures the system for you and provides up to 614K of memory.

Re-configure to the minimum size, e.g. *CONFIGURE FontSize 0. I change the configuration as follows:

```
Fontsize 0,     RamFsSize 0,
RMASize 0,      Screensize 10,
Spritesize 0,   SystemSize 0, Mode 0.
```

Then press Ctrl-Break and run the Emulator. When I have finished using the Emulator I restore the system to its original state (using *RMREINIT).

To simplify the above procedures you can create two EXEC files called START and END which contain the appropriate set of *UNPLUG, *CONFIGURE and *RMREINIT commands. Begin a session by pressing Reset, *EXEC START, press Ctrl-Break, and then run the Emulator. After a PC session press Reset, *EXEC END and press Ctrl-Break. This procedure can be automated by using a re-configuration program like that in RISC User Issue 6.

## INTO MS-DOS

Acorn has supplied MS-DOS version 3.21 with the Emulator. As MS-DOS loads it looks for two files on the disc, CONFIG.SYS and AUTOEXEC.BAT. CONFIG.SYS sets the configuration (of the MS-DOS system) for the session, and once set cannot be changed without rebooting. AUTOEXEC.BAT contains a list of commands to be performed on startup. I will explain more about these files in a later article. Neither file is essential to get MS-DOS started. If CONFIG.SYS does not exist a default configuration is used. If AUTOEXEC.BAT is missing MS-DOS will ask for the date and time; on the Archimedes you only need to press Return to get these values from the Archimedes' own clock.

## BACKING UP THE EMULATOR DISC

It is desirable to make back-up copies of both the discs supplied with the Emulator package. The PC Emulator disc is the easiest as it is in ADFS format and can be copied to a new disc using the standard ADFS *BACKUP command (*BACKUP 0 0 Q with a single drive, *BACKUP 0 1 Q with dual drives).

## BACKING UP THE MS-DOS SYSTEM DISC

The MS-DOS system disc is in MS-DOS format and cannot be backed up under the Archimedes operating system. We must first load the Emulator and MS-DOS disc. Once you see the MS-DOS prompt, A>, which is equivalent to the Archimedes * prompt, you can back up the MS-DOS disc. First set the write-protect notch (for safety), and then re-insert the system disc in drive A. Note that MS-DOS refers to drive 0 as drive A and drive 1 as drive B. Type DISKCOPY and press Return.

DISKCOPY is similar to Acorn's *BACKUP command. The source disc is, in this case, the MS-DOS disc. The target disc does not need to have been pre-formatted as this will occur automatically if necessary. If you have two disc drives, A and B, you can backup from A to B by the command "DISKCOPY A: B:". Note the colons after the drive letters - standard in MS-DOS. For more information refer to the MS-DOS section of the notes supplied with the Emulator, or a book such as *The MS-DOS Handbook* by R.King, published by Sybex. RU

# Archimedes Visuals

**More stunning visual effects to create on your Arc.**

## PIXEL-BASED 3D SPHERES

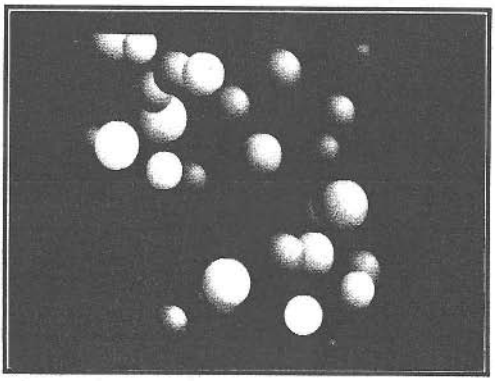
*by Mark Davis*

In RISC User Issue 1 we published a program created at Acorn for generating 3D spheres. The major weakness of this was that the Archimedes' 4096 colours with their 16 possible grey levels are insufficient to give a smooth shading effect. Even from a distance, bands of different shades are noticeable on the sphere, and spoil the effect. The way to get around the problem is to use a technique called *dithering*. This introduces a random element in the shade chosen for any given pixel within an object, and so blurs the edge between adjacent areas of shading. This technique was used in Issue 2 to draw a set of cylinders (again, courtesy of Acorn).

The program in listing 1 applies a similar technique to the drawing of spheres. Of course, since each pixel must be shaded separately, the routine is relatively slow, taking 10 seconds or so to draw a sphere of around 100 graphics units in diameter. The result however is extraordinarily good, with an excellent 3D feel to it. If you run the program in listing 1, a series of spheres of random colour and radius will be displayed. The dithering is achieved by the RND(1) component of the colour number set up in line 290. To see the effect of drawing the sphere without dithering, try removing RND(1) from the line. The result is a sphere distinctly marked with ridges.

**Listing 1**

```
 10 REM            >BasSphere4
 20 REM Program    Basic Spheres
 30 REM Version    A 0.4
 40 REM Author     Mark Davis
 50 REM RISC User  July 1988
 60 REM Program    Subject to Copyright
 70 :
 80 REM 256-colour modes
 90 REM col%=colour number
100 REM rad%=Radius
110 REM L1% & L2% Give light position
120 REM pix%=Pixel Size
130 :
140 MODE15:GCOL 149:CLG
```



```
150 REPEAT
160 ORIGIN RND(1279),RND(1023)
170 Z%=(RND(3)-1)+(RND(3)-1)*4+(RND(3)
-1)*16
180 PROCsphere(Z%,60+RND(25),30,30,2)
190 UNTIL FALSE
200 END
210 :
220 DEFPROCsphere(col%,rad%,L1%,L2%,pi
x%)
230 FOR Y%=rad% TO -rad% STEP -4
240 A%=(SQR(rad%*rad%-Y%*Y%)DIV pix%)*
pix%
250 FOR X%=-A% TO A% STEP pix%
260 P1%=DEG ASN(X%/rad%)
270 P2%=DEG ASN(Y%/rad%)
280 D1=ABS(P1%-L1%):D2=ABS(P2%-L2%)
290 C%=7.99-SQR(D1*D1+D2*D2)/14-RND(1)
300 IF C%<0 THEN C%=0
310 GCOL0,col%+(C% AND 4)*5.25 TINT(C%
AND 3)*64
320 PLOT69,X%,Y%:NEXT:NEXT
330 ENDPROC
```

As you can see from the listing, all the work is done by PROCsphere. This has five parameters: the colour number, the radius (in graphics units), two angular components of the incident light direction, and the horizontal pixel size (2 for modes 12, 15 and 20). The colour number is made up as follows:

```
col%=red+4*green+16*blue
```

where *red*, *green* and *blue* take integer values

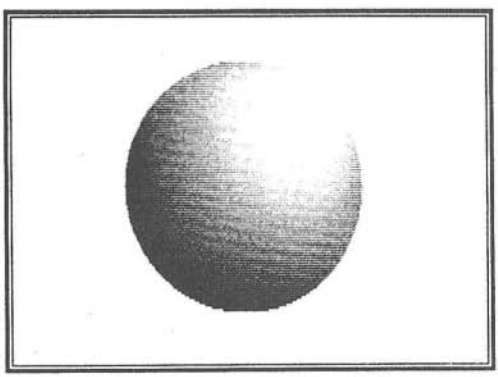
between 0 and 2, and represent the proportions of colour in the darkest part of the sphere. The procedure uses 8 different shades of the chosen colour, making full use of TINT. Each sphere is drawn at co-ordinates 0,0, and its physical position on the screen is determined by first setting the graphics origin to the desired position with ORIGIN x,y.

One special feature of this program is that the user can specify the position of the light source. This is achieved through the two parameters L1% and L2%. These represent the angle in degrees between the viewing angle and the normal to the surface at its brightest point. Thus with both set to 0, the brightest point will appear dead centre. With the first set to 30, and the second to 50, the sphere will be brightest at 30 degrees to the right of the sphere, and 50 degrees above it. Angles entered in this way may range from -180 to +180 degrees. A little experiment will clarify how it works.

Because of the usefulness of this procedure we are also giving a version suited to mode 12. This appears in listing 2. The parameters are the same as before, except that no colour number is passed. The routine now uses colours 8 to 15, which are defined in the early lines of the program. If you want to see an image of impeccable quality, run listing 2 in mode 20 (using a multi-sync monitor), and alter line 240 so that the Y step size is -2 rather than -4. This sphere will bowl you over!

**Listing 2**

```
  10 REM             >Mol2Sphr2
  20 REM Program     Mode 12 Spheres
  30 REM Version     A 0.4
  40 REM Author      Mark Davis
  50 REM RISC User   July 1988
  60 REM Program     Subject to Copyright
  70 :
  80 REM 16-colour modes
  90 REM rad%=Radius
 100 REM L1% & L2% Give light position
 110 REM pix%=Pixel Size
 120 :
 130 MODE12
 140 COLOUR 1,128,128,144
 150 GCOL 129:CLG
 160 FOR A=8 TO 15
 170 COLOUR A,A*16,A*16-128,A*16-128
 180 NEXT
 190 ORIGIN 640,512
 200 PROCmol2sphr(200,30,30,2)
 210 END
 220 :
 230 DEFPROCmol2sphr(rad%,L1%,L2%,pix%)
 240 FOR Y%=rad% TO -rad% STEP -4
 250 A%=(SQR(rad%*rad%-Y%*Y%)DIV pix%)*
pix%
 260 FOR X%=-A% TO A% STEP pix%
 270 P1%=DEG ASN(X%/rad%)
 280 P2%=DEG ASN(Y%/rad%)
 290 D1=ABS(P1%-L1%):D2=ABS(P2%-L2%)
 300 C%=7.99-SQR(D1*D1+D2*D2)/14-RND(1)
 310 IF C%<0 THEN C%=0
 320 GCOL0,C%+8
 330 PLOT69,X%,Y%:NEXT:NEXT
 340 ENDPROC
```



See *Animating Archie* in this issue and the next for applications of the sphere procedure. In addition to the version listed here, the magazine disc also contains a version of the sphere plotting procedure written in ARM assembler.

## SMART SCREEN CLEARING

*By Barry Christie*

Listing 3 gives a procedure for clearing any screen to a given colour using a special multi-sweep vertical bar effect. The program also contains a demonstration. The procedure is

**Continued on page 31**

# COLOURFUL PRINTING

**David Spencer gets printing on the Integrex Colourjet 132, and reports on his findings.**

One of the features of the Archimedes is its ability to display up to 4096 different colours on the screen, and we are becoming accustomed to the quality and sophistication of the screen displays that can be produced. What such a computer needs is a colour printer which will match the screen in its printed results. The Integrex 132 is just such a machine (if you've got £650 to spend), and its results are truly staggering when seen for the first time.

| | |
|---|---|
| Product | Integrex Colourjet 132 |
| Supplier | Integrex Ltd.<br>Church Gresley,<br>Burton on Trent,<br>Staffordshire DE11 9PT.<br>Tel. (0283) 215432 |
| Price | £654.35 Inc. VAT. |

When you first see the printer you could be forgiven for thinking it is yet another Epson FX80 lookalike. From the outside it looks nothing special, being very similar to other printers such as the Kaga Taxan, and Canon machines. In common with most printers, the Integrex will accept single sheets of paper, but unlike other printers it cannot be used with fan-fold paper. Instead, a roll of special high quality paper is used, this being placed under a flap on the top of the case. There is also provision to use overhead projector (OHP) film, allowing very attractive OHP slides to be produced given the appropriate software.

The Integrex 132 works by printing just four basic colours - black, yellow, cyan, and magenta. The last three are the so called 'secondary colours', and it is by mixing these that other colours can be produced. For example, a mixture of yellow and cyan will appear green on the paper. Further colours can be produced by printing adjacent dots in different colours. The small size of the dots fools the eye into seeing this as one colour.

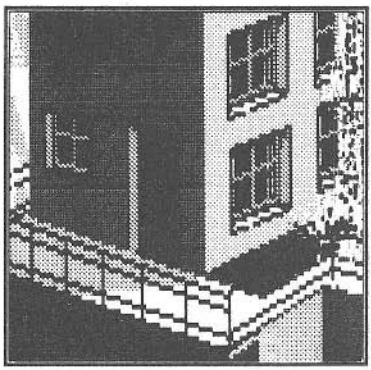
While most printers produce an image by striking pins against a ribbon, this technology does not emit colour printing particularly well. Although some printers do use a four colour ribbon, the resulting image is generally poor and colour mixing unsatisfactory.

The Integrex uses an inkjet technique, in which liquid ink is actually squirted onto the paper to produce a dot. This is achieved by a print head containing four nozzles side by side (one for each colour), and some nifty technology to control the ink flow. The actual ink is distributed from two cartridges, one containing black, and the other equal amounts of cyan, yellow, and magenta. When exhausted, the cartridges are replaced by opening a flap at the front of the printer, pulling out the old ones and plugging in new ones. The cartridges are claimed to last for over 4 million characters, more than most ribbons, but the replacement cost is £25 for the pair.

The Integrex 132 comes with a standard Centronics parallel interface, although there is an optional interface which allows both serial RS232 input, or a direct input from a Viewdata terminal. It is perfectly possible to just plug the printer into your Archimedes, connect it to the mains, and use it as a normal text printer. The drawback with this is that as only one dot can be printed at a time, printing is about a tenth of the speed of the average FX80 clone. This is further exaggerated by the lack of a sizeable buffer on the Integrex. Needless to say, using the Integrex as a text printer does it no justice at all.

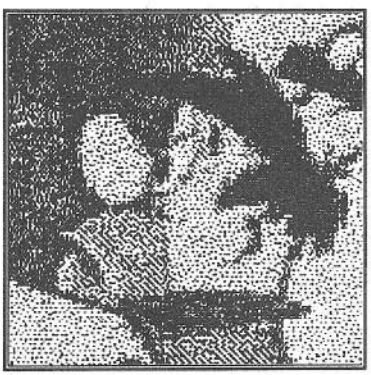
The real power of the Integrex only becomes apparent when you use a suitable screen dump program to generate reproductions of screen images. Despite all its elegance and

sophistication, the Integrex relies heavily on the quality of the screen dump software when it comes to printing out a copy of the screen. Clares' Artisan Support Disc (£19.95 inc. VAT, see RISC User Issue 6) includes an excellent dump for mode 12 only, and the printouts produced by this really do match the screen almost perfectly. A mode 15 dump has recently become available from Musbury Consultants (£30 inc. VAT), although the results we have so far achieved have been disappointing. We understand that Clares is also working on a screen dump for the 256 colour modes, but there is no information yet on availability or price.



Sample printout from Clares' screen dump.

Like most printers, the Integrex 132 can be controlled directly by Escape sequences, though those dealing with text (italics, bold etc.) are not Epson compatible. This may prove a limitation, but commercial software increasingly is making provision to support the Integrex 132 as an option.



Sample printout from Musbury Consultants' screen dump.

Another interesting feature of the Integrex 132 is its ability to operate in a Viewdata mode. In this mode the printer responds directly to the Escape sequences used by level 1 Videotext systems. This means that the data used to produce a teletext or viewdata page can be sent straight to the printer, and the page will be reproduced perfectly in full colour.

The documentation for the printer comes in the form of a 62 page A4 manual, which is dot matrix printed and fairly poorly bound. The manual covers the setting up of the printer, everyday use, and the various control sequences. There are appendices on the character set and the RS232 interface mentioned earlier. Printer manuals are often criticised for using too much technical jargon, making them particularly hard for beginners to understand, but the Integrex manual is even worse than usual. For the experienced programmer the manual is a fine source of reference material, but for the non-technical user who just wants to find out how to produce a certain special effect, it fails totally.

## CONCLUSION

The Integrex 132 is a superb colour printer, and anybody seeing the screen dumps of which it is capable will immediately fall in love with it! At the same time it is not cheap, but there is nothing else around which will so realistically reproduce an Archimedes colour screen on paper. During extensive use in the RISC User office the printer performed faultlessly, our only quibble being over-inking, which was easily cured by using a higher quality paper (this is £16 for a 25m roll from Integrex, or £5 a roll from Clares). It is a great pity that the manual is so poor when compared to the hardware - extra effort here by Integrex would have paid dividends. I can highly recommend the Integrex 132 for anybody who can justify the cost.

The screen dump programs mentioned are available from:

**Clares Micro Supplies, 98 Middlewich Road, Northwich, Cheshire CW9 7DA, (0606) 48511 and,**
**Musbury Consultants, 8 Fairhill, Helmshore, Rossendale, Lancashire BB4 4JX, (0706) 216701.**

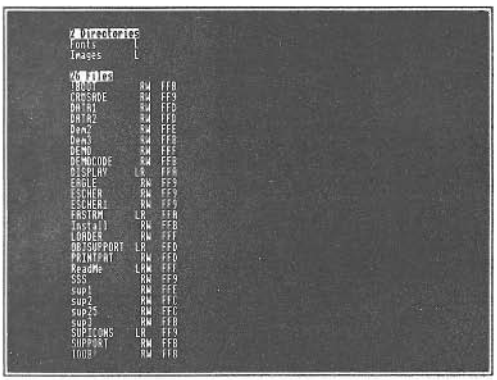
RU

# READING FROM THE ADFS

by Lee Calcraft

**It is often necessary to read data from the ADFS. We present here a procedure which performs this task, and which works equally well with the Econet filing system.**

There are many occasions when it is useful, or even essential, to get information back from the disc filing system. On the old Model B this was often an uphill struggle. But on the Archimedes Acorn have implemented more powerful system calls which make life a lot easier. Nevertheless, it is still necessary to set up RAM buffer areas, and to make operating system calls.




In this article, we have packaged the operation into a simple procedure. This is defined from line 390 onwards in the accompanying listing. The remainder of the program simply provides an illustration of its use. When you run the program, you will see a display similar to that in the accompanying screen shot, giving catalogue information for the currently selected directory. Files and sub-directories are listed separately for clarity, and both are accompanied by indicators to access status ("L", "W" and "R", as appropriate). File types are also displayed.

All this information is gathered from a *single* SYS call performed just once in line 420. The call returns a mass of data on the currently selected directory in a single block of RAM (at discbuff). But the procedure processes this information and stores it in a more useable way. After PROCreaddisc has been executed, the number of directories and files are held in the two variables dircount and filecount, the directory names in the array dir$(), and the filenames in the array file$(). Information about the objects is held in arrays dir(), and file().

The purpose of the code between lines 130 to 310 is to extract and display this information. Sub-directories are displayed by printing the contents of array dir$(n) in a loop from 0 to dircount-1 (lines 170 to 210). A similar operation on the array file$(n) in a loop from 0 to filecount-1 (lines 260 to 300), displays filenames. To display access status, the program prints FNaccess for each object. Finally the file type is displayed using FNtype.

If you are going to use PROCreaddisc in other applications, you may well wish to alter its two calling parameters. These are dirname$ and wildcard$ respectively. The first defines the object name of the directory to be read. In the example given we have set this to "", causing the currently selected directory to be interrogated, but we might equally well have set it to "$.PROGRAMS.APPLICATIONS", or whatever. The second parameter is the wildcard for the directory interrogation. We have used "*" to force all objects in the parent directory to be recorded, but we could equally well have used "PROG*", for example, which would only record directory and filenames beginning with the letters "PROG".

**ECONET:** The only change to the program required for use on Econet is to alter the four occurrences of the number 77 to 147 in lines 100 and 110 to allow for the larger number of objects per directory.

```
10 REM                >ADFSutil6
20 REM Program        Reading the ADFS
30 REM Version        A 0.6
```

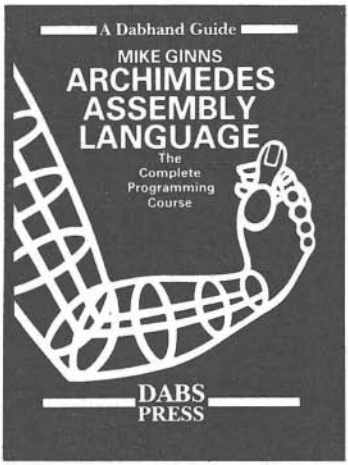
```
   40 REM Author    Lee Calcraft
   50 REM RISC User July 1988
   60 REM Program   Subject to Copyright
   70 :
   80 DIM discbuff &1000
   90 DIM dirname &100,wildcard &20
  100 DIM dir$(77),dir(77,1)
  110 DIM file$(77),file(77,1)
  120 :
  130 PROCreaddisc("","*")
  140 MODE0:VDU23,17,5|
  150 PRINT ;dircount;" Directories";
  160 VDU23,17,5|:PRINT
  170 IF dircount>0 THEN
  180 FOR N=0 TO dircount-1
  190 PRINTdir$(N) TAB(12)FNaccess(dir()
)
  200 NEXT
  210 ENDIF
  220 PRINT
  230 VDU23,17,5|
  240 PRINT;filecount;" Files";
  250 VDU23,17,5|:PRINT
  260 IF filecount>0 THEN
  270 FOR N=0 TO filecount-1
  280 PRINTfile$(N) TAB(12)FNaccess(file
()) TAB(17)FNtype(file())
  290 NEXT
  300 ENDIF
  310 END
  320 :
  330 DEFFNaccess(array())
  340 =CHR$(32+5.5*(array(N,1) AND 8))+C
HR$(32+50*(array(N,1) AND 1))+CHR$(32+27
.5*(array(N,1) AND 2))
  350 :
  360 DEFFNtype(array())
  370 =STR$~(array(N,0) AND &FFF)
  380 :
  390 DEFPROCreaddisc(dirname$,wildcard$
)
  400 $dirname=dirname$+CHR$(0)
  410 $wildcard=wildcard$+CHR$(0)
  420 SYS "OS_GBPB" ,10,dirname,discbuff
,77,0,&2000,wildcard TO , , ,objects
  430 addr=discbuff
  440 dircount=0
  450 filecount=0
  460 IF objects>0 THEN
  470 FOR n=0 TO objects-1
  480 PROCgetname
  490 NEXT
  500 ENDIF
  510 ENDPROC
  520 :
  530 DEFPROCgetname
  540 type=addr?&10
  550 IF type=1 THEN
  560 PROCgetit(file$(),file(),filecount
)
  570 filecount+=1
  580 ELSE
  590 PROCgetit(dir$(),dir(),dircount)
  600 dircount+=1
  610 ENDIF
  620 ENDPROC
  630 :
  640 DEFPROCgetit(obj$(),obj(),index)
  650 obj(index,0)=((!addr)>>>8) AND &FF
F
  660 obj(index,1)=addr?&0C
  670 addr+=&14
  680 objname$=""
  690 zero=FALSE
  700 REPEAT
  710 byte=?addr
  720 IF byte=0 THEN
  730 zero=TRUE
  740 ELSE
  750 objname$+=CHR$(byte)
  760 ENDIF
  770 addr+=1
  780 UNTIL zero
  790 obj$(index)=objname$
  800 overhang=addr MOD 4
  810 IF overhang>0 THEN addr+=4-overhan
g
  820 ENDPROC
```

*Next month we will publish a very short routine which calls PROCreaddisc recursively, and displays all occurrences of a supplied wildcarded filename (together with their paths from the root). This is very useful for tracking down lost files on a disc.*

# ARCHIMEDES ASSEMBLY LANGUAGE

**David Spencer reviews Dabs' eagerly awaited guide to ARM assembler.**

When the Archimedes was launched about a year ago, it was more than just a new computer, because the microprocessor used in it was also brand new. The information supporting this machine, for this reason, has been fairly sparse. To date the only book available has been Peter Cockerell's *ARM Assembly Language*, which was published before the Archimedes itself was released. As a result, this book does have some drawbacks, principally the lack of any detail relating to the Archimedes, although there is a chapter on the BBC Basic assembler.




Archimedes Assembly Language
by Mike Ginns, published by Dabs Press at £14.95.

Apart from that, those who have wanted to learn the Archimedes assembly language have really had to rely on information given in RISC User (see our current series on ARM assembler), and other magazines to supplement the meagre information given in the *User Guide* and the *Programmer's Reference Manual*.

*Archimedes Assembly Language - A Dabhand Guide* is a new book that fills this obvious gap, though it probably won't be the last. *Archimedes Assembly Language* is a 368 page tome, divided into 24 chapters and 11 appendices. Like the *User Guide*, this book is wire bound so that it can be laid flat when open. In its introduction the book claims to be able to teach the beginner how to program the Archimedes in assembly language. This is not an easy task to undertake with any computer, and ARM assembler is probably more difficult than most.

The book starts off with a chapter explaining the architecture of a typical computer system. While this does provide a firm basis for the study of assembly language, I feel that somebody relatively new to the subject will find it rather daunting, and a more relaxed introduction would have been better. However, the chapter does provide some useful information, and is certainly worth reading.

This is followed by chapters on the internals of the ARM chip, and the Basic assembler. While these opening chapters are well written, they tend to lay down a wealth of hard facts, but give few examples, and the book is rather the worse for this.

The next section, which covers eight chapters, deals with the instruction set of the ARM processor. This begins with an overview of the various instruction types, and then goes on to explain each in turn. A chapter is devoted to the barrel shifter, a concept which causes problems to many beginners, and another to the use of the program counter as an instruction operand. Again, this can be hard to understand. The book explains both these topics in a way that should clear up any confusion.

The final chapter in this section deals with the use of stacks, covering the different types and how they are set up. Some of the possible uses of stacks are also covered. Once again, this section seems a bit dry, and lacks examples. The ordering of topics also results in some subjects being referred to before they have been covered.

The next two chapters deal with the advanced features of the Basic assembler, and techniques for assembling and debugging

programs. This includes information on the implementation of macros and conditional assembly, and also full instructions for using the Debugger.

The rest of the book moves away from the fundamentals of ARM assembly language and onto more advanced topics. The first chapter in this section looks at interrupts and events. This only gives a brief introduction, and Acorn's *Programmer's Reference Manual* is more helpful once you understand the basics. I just wish somebody would come up with an analogy with which to explain interrupts other than an office worker answering the phone. The next chapter handles vectors, including both the hardware vectors used by the ARM processor (for such tasks as interrupt handling and address exceptions), and also the software vectors used by the Arthur operating system. Yet again only a brief introduction is given and the *Programmer's Reference Manual* provides more detail.

Chapter 17 gives details of some of the SWI calls that Arthur provides. In a book of this nature it is clearly not possible, nor even desirable, to cover the very large number of SWI calls that are available on the Archimedes, and a selective choice must be made. The chapter starts off by explaining the various calls connected with character and string input and output, and then moves on to number conversions, and other calls such as OSBYTE and OSWORD.

The next couple of chapters cover the complex subjects of the window manager and the font system in ARM assembler. Both of these topics can cause major problems to beginners, and while they can't be covered in full in a book such as this, a good introduction is provided. However, it is only when this point is reached that the book introduces any practical and interesting examples of programs in ARM assembler.

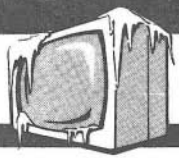
Although much of this book seems to provide a heavy diet of factual information, this is partly rectified in the final section. The last five chapters show how to implement various Basic functions and statements in ARM assembler. The author handles input/output, string manipulation, functions, operators, control statements, loops and graphics, giving what he calls 'templates' for all the various structures. These are basically skeleton assembler programs to use in your own code. I suspect that a relative beginner to assembly language would find this one of the most useful sections.

The appendices, which occupy a total of 41 pages, start off with sections on numeric representation, arithmetic and logic operators. The format of ARM instructions at a bit level is described, and the appendices finish off with lists of SWI, OSBYTE and OSWORD calls, VDU codes and PLOT numbers. One omission is a table of the execution times for each ARM instruction. However, with the ARM being a pipelined processor, the usefulness of such information is doubtful.

The style of the text throughout the book is easy to read, and the layout is uncluttered. Good, clear diagrams are used where appropriate to make explanations easier, and the book's comprehensive index makes finding any piece of information a painless process.

All the example programs given in the book are also available on disc (at the price of £9.95 inc. VAT). In addition, there are a number of very useful programs on the disc which are not listed in the book, including a memory and disc sector editor, and a complete scrolling disassembler.

Mike Ginn's book takes a quite formal approach to the ARM assembly language, and is one which beginners might find hard going compared with other similar books which adopt a more tutorial style. I still believe it to be better than Peter Cockerell's book, particularly for the more experienced machine-code programmer, because it is so much more specific to the Archimedes. Subject to these reservations, I would recommend *Archimedes Assembly Language*, but if you are a beginner don't expect it to be easy going. RU



# SCREEN FREEZER AND DUMPER

by Nic van Someren

**Use this relocatable module to freeze any program at any point, and optionally dump the current screen to disc.**

It is sometimes useful to be able to freeze a program at will, and to be able to dump screens to disc at any point. The relocatable module listed here allows you to do this under most circumstances. To make use of it, first type it in and save it to disc. When you run it, a relocatable module called FreezerRM will be saved to disc. To use this at any time, load it into the RMA using:

```
*FreezerRM
```

Once the module is installed, you can check its presence with:

```
*HELP MODULES
```

The module provides a single command:

```
*Freeze
```

and you can obtain information about it with:

```
*HELP Freeze
```

Once you have issued the command *Freeze, the module will check every keyboard entry to see if both Alt keys are pressed simultaneously. If so it will freeze the current program, and wait for a keypress. If "S" is pressed it will save the current screen using:

```
*SCREENSAVE ScreenNo01
```

Alternatively, if any other key is pressed it will resume the original program. If you save further screens, it will increment the filename used to ScreenNo02, and so on.

The program works by using interrupts. The code makes use of Arthur's so-called "callback" facility in responding to the key-pressed event, and is disabled during operating system calls. As a consequence you will find some circumstances under which pressing Alt-Alt has no effect (e.g. during the GET function). Also, if you are running software which alters the interrupt vectors, then the freezer will not work.

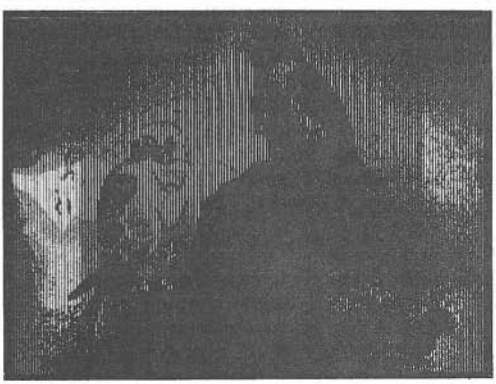
```
 10 REM             >Freeze7
 20 REM Program     Screen Freezer
 30 REM Version     A 0.7
 40 REM Author      Nic  van Someren
 50 REM RISC User   July 1988
 60 REM Program     Subject to Copyright
 70 :
 80 DIM code% 1000
 90 FOR pass%=4 TO 7 STEP 3
100 P%=0:O%=code%
110 [OPT pass%
120 EQUD 0:EQUD init
130 EQUD finish:EQUD 0
140 EQUD title:EQUD helpstring
150 EQUD commandhelp:EQUD 0
160 EQUD 0:EQUD 0:EQUD 0
170 .title
180 EQUS "Freezer"
190 EQUB 0
200 ALIGN
210 .helpstring
220 EQUS "Freezer"
230 EQUB 9:EQUB 9
240 EQUS "1.00 (17 Jun 1988)"
250 EQUB 0
260 ALIGN
270 .commandhelp
280 EQUS "Freeze"
290 EQUB 0
300 ALIGN
310 EQUD init:EQUD 0:EQUD syntax
320 EQUD help:EQUD 0
330 .help
340 EQUS "*Freeze will freeze the scre
en whenever both Alt keys are pressed to
gether. Then press S to save the screen,
 or any other key to unfreeze."
350 EQUB 13
360 .syntax
370 EQUS "Syntax : *Freeze"
380 EQUB 0
390 ALIGN
400 .init
410 STMFD R13!,{R14}
420 ADR R0,saveblock
430 ADR R1,callbackcode
440 SWI "OS_CallBack"
450 ADR R2,oldcallback
460 STMIA R2,{R0,R1}
470 MOV R0,#&10:ADR R1,eventcode
480 MOV R2,#0:SWI "OS_Claim"
490 MOV R0,#14:MOV R1,#11
500 SWI "OS_Byte"
510 LDMFD R13!,{PC}
520 .finish
530 STMFD R13!,{R14}
540 MOV R0,#13:MOV R1,#11
550 SWI "OS_Byte":MOV R0,#&10
560 ADR R1,eventcode
570 MOV R2,#0:SWI "OS_Release"
580 ADR R2,oldcallback
590 LDMIA R2,{R0,R1}
600 SWI "OS_CallBack"
610 LDMFD R13!,{PC}
```

```
620 .oldcallback
630 EQUD 0:EQUD 0
640 .eventcode
650 CMP R0,#11:MOVNE PC,R14
660 STMFD R13!,{R0-R3,R14}
670 MOV R0,PC:ORR R3,R0,#3
680 TSTP R3,#3:MOVNV R0,R0
690 STMFD R13!,{R0,R14}
700 ADD R2,R2,#2:BIC R3,R2,#2
710 CMP R3,#&60:BNE notourkey
720 AND R2,R2,#2:MOV R3,#1
730 CMP R1,#1:LDR R0,keyflags
740 ORREQ R0,R0,R3,LSL R2
750 BICNE R0,R0,R3,LSL R2
760 STR R0,keyflags
770 CMP R0,#5:SWIEQ "OS_SetCallBack"
780 .notourkey
790 LDMFD R13!,{R0,R14}
800 TSTP R0,R0:MOVNV R0,R0
810 LDMFD R13!,{R0-R3,PC}
820 .keyflags
830 EQUD 0
840 .callbackcode
850 SWI "OS_IntOn":SWI "OS_ReadC"
860 CMP R0,#ASC"S":BNE nosave
870 LDR R0,screennumber
880 ADD R0,R0,#1
890 STR R0,screennumber
900 ADR R1,screennameend
910 MOV R2,#3
920 SWI "OS_ConvertHex2"
930 MOV R0,#2
940 ADR R2,screenname
950 MOV R3,#1
960 SWI "XOS_SpriteOp"
970 .nosave
980 SWI "OS_IntOff"
990 ADR R14,saveblock
1000 LDMIA R14,{R0-R14}^
1010 LDR R14,[R14,#15*4]
1020 MOVS PC,R14
1030 .saveblock
1040 EQUS STRING$(64," ")
1050 .screennumber
1060 EQUD 0
1070 .screenname
1080 EQUS "ScreenNo"
1090 .screennameend
1100 EQUS "XX":EQUB 0
1110 ALIGN
1120 ]
1130 NEXT
1140 OSCLI"SAVE FreezerRM "+STR$~code%+
" "+STR$~O%
1150 OSCLI"SETTYPE FreezerRM FFA"
```

RU

## *ARCHIMEDES VISUALS (continued from page 23)*

called after setting the colour for the clearance with GCOL (line 80). Its only parameter (currently 4) is simply twice the horizontal pixel size of the mode in use.



### Listing 3

```
 10 REM >ClearL1
 20 REM Fancy Clear
 30 REM By Barry Christie
 40 :
 50 MODE12
 60 GCOL7:RECTANGLEFILL200,100,800,800
 70 PROCwait(100)
 80 GCOL1:PROCclear(4)
 90 END
100 :
110 DEFPROCclear(step)
120 FOR X%=0 TO 1280 STEP step*2
130 PLOT4,X%,0:PLOT5,X%,1023
140 PLOT4,1278-X%,0:PLOT5,1278-X%,1023
150 PROCwait(1)
160 NEXT
170 PROCwait(70)
180 FOR X%=step TO 1280 STEP step*2
190 PLOT4,X%,0:PLOT5,X%,1023
200 PLOT4,1278-X%,0:PLOT5,1278-X%,1023
210 PROCwait(1)
220 NEXT:ENDPROC
230 :
240 DEFPROCwait(T)
250 TIME=0:REPEAT UNTIL TIME=T
260 ENDPROC
```

RU

# INTRODUCING ARM ASSEMBLER (4)

by Lee Calcraft

**This Month: Memory Access and Searching for Strings**

## LOADING ARM REGISTERS

There are two quite distinct types of instruction which can be used for putting data into ARM registers: the **MOV** family of instructions and the **LDR** family. We have used examples of both in earlier programs in this series, though without discussing them in any detail. They are in practice very different types of instruction, and are used in different ways.

Essentially the MOV instruction is used for transferring data from one ARM register to another. Thus:

```
MOV R0,R1
```

will *move* the contents of register R1 into R0. Like all ARM instructions it can be made conditional. The following:

```
MOVCS R0,R1
```

will only perform the move if the carry flag is set. Similarly we can use the "S" suffix to ensure that the flags are set according to the result of the instruction. Thus:

```
MOVS R0,R1
```

will place the contents of R1 into R0, and set the Z and N flags according to the new contents of R0. Other flags remain unaltered.

The MOV instruction may also be used to load any register with immediate data in the following way:

```
MOV R0,#255
```

This will place the value 255 into R0. As before, the instruction can take both the conditional and flag setting suffixes, as required. There is however, one extremely important proviso to the use of MOV with an immediate operand, which must always be borne in mind. All ARM instructions are exactly one 32 bit word in length. There is thus not sufficient room in the MOV instruction word for very large numbers. In fact there is room for just 12 *bits* of data. This would be large enough to represent integers in the range 0 to 4095. However, the designers of the chip at Acorn decided to allocate the 12 bits in a more subtle, and inherently more useful way. Eight bits are allocated to an integer value, and the remaining four are used to specify the degree of shift (or more precisely, of rotation) to be applied to the accompanying integer.

Fortunately, the Arc's built in assembler works it all out, and the programmer need never know what is going on. Thus if you try the following:

```
MOV R0,#&10000
```

the assembler will automatically break the number down into an immediate and a rotated component, and continue with the assembly. Of course, not all numbers can be represented in this way, and if the assembler comes across one which cannot be suitably expressed, it will report the error with the following message:

```
Bad immediate constant at line xxx
```

If you wish to load into ARM registers constants which cannot be broken down in this way (eg. &101 etc.) you will need to place the data into RAM using the EQU family of assembler directives, and then load the register or registers from RAM using variants of the LDR instruction.

## THE LDR/STR FAMILY OF INSTRUCTIONS

The LDR/STR group of instructions is quite distinct from variants of MOV. And although this group can be used with both conditional and "S" suffixes, it differs from them in a number of ways. For example, LDR (Load Register) and STR (store contents of Register) are used only for transferring data between registers and RAM, and may not be used with immediate operands. In its simplest form the pair of instructions:

```
LDR R0,[R1]
STR R0,[R2]
```

will load register R0 with the 32 bit word located at the address held in register R1, and will store it at the address held in R2.

Because of hardware constraints, the word must be located at a word boundary to ensure a correct load (or save). In other words the address must be divisible by four; or put another way, the bottom two bits of the address must be zero. If the load is carried out from a non word boundary, the processor ensures that

at least the bottom byte of the loaded value is correct, though the reliability of the top three bytes of the loaded word depends on its exact position relative to a word boundary. For further detail on this, see Cockerell, *ARM Assembly Language Programming*, p.59.

If you are writing a program which involves reading in long sequences of data from RAM, which may not start at a word boundary, the most efficient approach is to load from the nearest word boundary, and adjust the data accordingly. In other circumstances it is easier to use a variant of the LDR instruction which loads data one byte at a time. The instructions LDRB and STRB load and save data one byte at a time irrespective of whether the given address falls at a word boundary or not. Thus:

```
LDRB R0,[R1]
STRB R0,[R2]
```

will copy the byte at the address held in R1 into the location whose address is held in R2. In the case of LDRB the top 3 bytes of the destination register are set to zero as an outcome of the load.

## INDEXED ADDRESSING

All four instructions LDR, STR, LDRB and STRB can take indexed addressing, which may either be of the pre- or post-indexed variety. To illustrate pre-indexed addressing, compare the three load instructions:

```
LDR R0,[R1]          Indirect
LDR R0,[R1,#&20]     Pre-indexed indirect
LDR R0,[R1,R2]       Pre-indexed indirect
```

In the first of these examples, R0 is loaded with the contents of RAM whose address is held in R1. For this reason it is called *indirect addressing*. In the second case R0 is loaded with the contents of the location whose address is given by adding &20 to the contents of R1. Because of the fixed length of ARM instructions this numeric offset is again strictly limited in magnitude. This time it may range from -4095 to +4095.

In the third case the address of the load is found by adding the contents of R1 and R2. A further complication permits us to specify that the contents of register R2 be shifted by a specified number of places before the load takes place. But we will defer discussion of this and of post indexed addressing to a future issue. In the mean time we will put some of the instructions which we have covered here to work in a string search routine.

## A GENERAL STRING SEARCH ROUTINE

The program in the accompanying listing is a general string search routine. It requires three inputs handled from Basic: a start and end address for the search (in hex), and a string to be searched for. If you press Return on a null input it will supply PAGE as the start address and HIMEM as the end address. Search strings may be up to the full 255 bytes allowed by Basic.

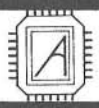
As it currently stands, the search is case specific. You may change this by inserting the following two instructions:

```
234 AND R5,R5,#&DF
236 AND R6,R6,#&DF
```

These force the bytes compared into upper case before each comparison is made (assuming that they are restricted to the range ASCII 65 to 122). The routine may also be easily modified to make it search for any match which the user requires. For example, to make it search for any sequence of hex bytes, just alter the Basic input routine to read in a sequence of bytes, and store these at *buffer*, setting B% to the number of bytes. Alternatively, to make it perform a 32 bit word search where words may be located across word boundaries, just modify the input routine to accept a 32 bit word, place this at *buffer*, and set B% to 4.

## HOW IT WORKS

The functions of the registers used by the routine are given in REM statements at the start of the program. You may well find that this is a useful practice to follow. Because the programmer has some 15 or so registers at his disposal it is very easy to get confused over which are used for which task. By specifying their functions at the start, you are forced to clarify the issue early in a program's development. In the present program we need three address pointers: one for the address of the supplied string (R0), one to point to the



current character in that string (R4), and one to point to the address in RAM of the sequence currently being tested (R2). We also need to keep a record of the length of the search string (R1), and of the highest address in RAM to be searched (R3). Finally we need two registers to hold the two bytes being compared at any time (R5 and R6).

When the routine is called, the initial contents of R0 to R3 are automatically set up by Basic (being the contents of variables A% to D% at the time that the machine code is called). Line 200 ensures that the pointer to the search string is set to zero, so as to point to its first byte. Then R4 is loaded with the first byte of the search string. Note the use of indexed addressing in the LDRB instruction to load from the address given by the base address of the string plus the character offset (initially zero). In line 230 a similar load takes place, but this time the first byte from the search area is loaded. The two are compared, and if they do not match, we start again at *mainloop*, resetting the character pointer to zero to point to the first character of the string, but incrementing the pointer to RAM (register R2) at line 330.

If, on the other hand, the two bytes match, we increment the character pointer (line 260) and compare the next two characters. A check is also made against the length of the search string (line 270), since if the full length of the search string has been checked, the whole string must match that found in RAM, and a match is reported (line 300). In this simple program we load R0 with 255 if a match is found, or with zero if not. Basic then simply checks the contents of R0 with the USR function (lines 540 and 550), and responds to the result.

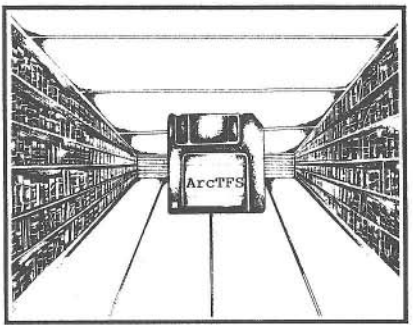
```
10 REM >4-1ARMpg4
20 REM String Search Routine
30 :
40 REM  R0 base addr of search string
50 REM  R1 length of search string
60 REM  R2 start address
70 REM  R3 End address
80 REM  R4 String offset pointer
90 REM  R5 Current string byte
100 REM  R6 Current RAM byte
110 :
120 DIM space 1000
130 DIM buffer 256
140 FOR pass=0 TO 1
150 P%=space
160 [
170 OPT pass*3
180 .start
190 .mainloop
200 MOV R4,#0;      Zero pointer
210 .nextletter
220 LDRB R5,[R0,R4];Load string byte
230 LDRB R6,[R2,R4];Load RAM byte
240 CMP R5,R6;      Compare
250 BNE mismatch;   Mismatch
260 ADD R4,R4,#1;   Match so next byte
270 CMP R4,R1;      End of string?
280 BNE nextletter;No
290 .match
300 MOV R0,#255;    Indicate match
310 B exit
320 .mismatch
330 ADD R2,R2,#1;   Inc RAM pointer
340 CMP R2,R3;      End of RAM?
350 BNE mainloop
360 MOV R0,#0;      Indicate fail
370 .exit
380 STR R2,result;  Store address
390 MOV PC,R14;     Return
400 :
410 .result
420 EQUD 0
430 ]:NEXT
440 :
450 REPEAT
460 INPUT'"Start address? &"A$
470 INPUT"End address? &"B$
480 INPUT"Search word ",C$
490 IF A$="" A$=STR$~PAGE
500 IF B$="" B$=STR$~HIMEM
510 C%=EVAL("&"+A$):D%=EVAL("&"+B$)
520 $buffer=C$
530 B%=LEN(C$):A%=buffer
540 Z=USR(start)
550 IF Z>0 PRINT"Found at &";~(!result
) ELSE PRINT"No find"
```

*Next month we will take a look at the multiple load and store instructions, and the associated implementation of subroutines.*

RU

# ArcTFS - THE RESEARCH & AUTHOR'S DATABASE

* Indexes, cross-references and files typed or imported ASCII text items up to 7700 characters in length.
* Keeps searchable alphasorted files of explainers for all indexing and source codes.
* Creates additional referencing fields to suit the subject.
* Creates text files up to 650K in length on a 800K disc - the equivalent of 80,000-100,000 words.
* Searches text items directly or any referencing field.
* Formats any subset and saves it as a file which will be read as a default - formatted native file, answerable to all reformatting commands, by ArcWriter or 1st Word Plus.
* Translates Wordwise or View files in ASCII form into ArcWriter or 1st Word Plus format, and gives database indexing and control of these files in the meantime.




*"I have no hesitation in recommending it to anyone engaged in research, or anyone else who needs to file a large number of disparate pieces of text. At the price it is excellent value for money...." Richard Seel, ST World, June 1988, writing of the Atari ST version. (The Archimedes version is considerably enhanced.)*

## ArcTFS (Text Filing System) - £29.95

(Cheque or PO only) to:

## TEXCELLENCE, 2 Greenhill Road, Coleraine, N.Ireland BT51 3JE

*(For further details send SAE)*

# PRESENTER

*AFFORDABLE PRESENTATION GRAPHICS FOR THE*

ARCHIMEDES

*PRESENTER* is a truly professional graphics program which allows the user to create, modify, print out or photograph high quality colour displays of data in either bar, pie or line format. Data is entered either manually in spreadsheet fashion or can be imported from other spreadsheets or programs in comma separated format. *PRESENTER* makes full use of the Archimedes' WIMP environment and high resolution graphic modes and incorporates 3D displays, auto-scaling axis, user-definable layouts and multi-layer graphs. Screens may be saved for use with graphics packages like *ARTISAN* or graphic wordprocessors like *1ST WORD PLUS* & *GRAPHIC WRITER*. *PRESENTER* can also be used with *PIPEDREAM*.

At only £24.95 ex-VAT, *PRESENTER* is an invaluable extension to existing wordprocessors and spreadsheets. It is a **must** for all Archimedes' users.

*PRESENTER* is availble from all good Acorn dealers, or direct from LINGENUITY. Telephone orders accepted - ring 098 685 476.

Please send me...*PRESENTER* Packs ..... Retail Price £24.95 £.....
VAT £ 3.74 £.....
Postage UK £ 1.15 £.....
Postage Overseas £ 6.00 £.....
TOTAL £.....

Payment Visa.... Access.... Cheque....

Credit card number .....
Card Expiry Date .....
Name as on credit card .....
Address .....
.....
.....Postcode.....

Send to LINGENUITY, Dept.AL, P.O.Box 10, Halesworth, Suffolk IP19 0DX.

Lingenuity
Specialist Software

# Postbag

*The increasing number of pages in RISC User has now allowed us to introduce a page for readers. If you have any response to the ideas expressed this month, or views or information on anything else related to the Archimedes then we would be delighted to hear from you.*

## PC EMULATOR QUERIES

**I read with interest the review of the 5.25" disc interface in RISC User Issue 5. I have an A440 with PC emulator version 1.0. My need is to read, write and format IBM 360K format 5.25" discs. Can you answer the following:**

**Will the described interface allow me to do this?**
**What would be a suitable drive?**
**Will the PC emulator need to be upgraded?**

**Thank you in advance for your help, and for an excellent magazine.**

**David Snell**

*This interface will do what is required, and we would recommend a 40 track double-sided drive (any reputable make) if you wish to* ***format*** *5.25" discs as described. If the requirement is just to read or write pre-formatted discs then a switchable 40/80 track double-sided drive should suffice.*

*It would be desirable to upgrade to the latest version of the PC emulator as this not only offers greater speed and more available memory than before, but is also more flexible in handling drives external to the Archimedes.*

*To format a 360K 5.25" disc, load MS-DOS and then issue the command:*

*FORMAT A: /4*

*where A: is the appropriate drive designation and '/4' indicates 360K size. The formatting can be checked with:*

*CHKDSK A:*

## RETROGRADE ROMS

**Am I the only one who thinks producing software for the Archimedes on ROM is a retrograde step? Software won't run directly from ROM on the Archimedes, but has to be loaded into RAM first. By using ROMs this incurs a hardware cost overhead of approximately £100 on the 300 series, as well as using up one of the valuable podule slots. All Archimedes have a disc drive; why not use it?**

**Michael Lowe**

*Why not indeed? It is good enough for the PC and Macintosh market. So far it would appear that only Computer Concepts are heavily committed to ROM software on the Archimedes. However, we have some sympathy here for CC. ROM based software is the only reliable method of software protection and CC's decision to use this may be seen as a reflection on the state of the Acorn market.*

## MEMBERS' ADS

**I had been a member of BEEBUG for several years - almost since year one - when you introduced RISC User. There are no private ads pages in RISC User, a fact I regret now that I have Beeb items for sale, but no longer subscribe to BEEBUG. Could I suggest that RISC User members are allowed to place adverts in BEEBUG until RISC User can provide this service.**

**J.Hodgkins**

*We are very happy to do this, and indeed RISC User members may find that BEEBUG is a more appropriate place to advertise unwanted BBC items anyway. Just quote your RISC User membership number, and mark the envelope 'Personal Ads'.*

## ARCWRITER

**I have heard rumours that there is a new version of ArcWriter. The current version seems to have quite a few bugs in it - text files allowed only on drive :0, and all files date stamped circa 1900 for example. It would also be nice to change the rather garish colour scheme. After MacWrite, ArcWriter seems extremely limited, even if it's 'free'.**

**Graham Allan**

*The offer to supply ArcWriter to new purchasers of the Archimedes has now terminated. Acorn says that there will be no further versions of ArcWriter and that the future lies with 1st Word Plus alone, as ArcWriter was never intended to be a long term product. Acorn is offering to trade in ArcWriter for 1st Word Plus which may then be purchased at the special price of £45.97. All registered Archimedes owners should have received a letter to this effect. For further information contact Acorn direct on 0223-214411.*

*The comparison with the Apple Macintosh is quite valid. If the Archimedes is the fastest micro in the world (faster than a Macintosh), it surely deserves software that matches that potential.* RU

## AUTO-INSTALL FOR SCREEN MANAGER ETC.

By Lee Calcraft

The accompanying EXEC file permits you to call the Screen Manager (Issue 5), or any other program, from immediate mode Basic without disturbing any resident program. It is also possible to call it from within another program, but more of that anon. It works by installing the Screen Manager in the Arc's library, and calling it from there. The only subtlety is that it makes a check to see whether it is already present before installing it.

It does this by first setting the error variable (ERR) to 4 by forcing an error (hence the "Z" in the EXEC file). It then calls an empty procedure (PROCsmantest) which has been added to the end of the Screen Manager (DEFPROCsmatnest:ENDPROC). If the error number is 29 (No such Proc), the Manager is installed into the library. Then, providing that the error number is either 4 or 29, PROCscreenman, the Manager's main procedure entry, can be called. Any other error number is reported in the normal way. When you have finished with the Screen Manager, just use the QUIT box, or press Return. The version of the EXEC file supplied assumes that the Screen Manager is saved as ScrnMan3 in the Library directory of a disc named "WorkDisc".

```
VDU21
Z
PROCsmantest
IF ERR=29 THEN INSTALL ":WorkDisc.$.L
IBRARY.SCRNMAN3"
IF ERR<>29 AND ERR<>4 THEN VDU6:REPOR
T:VDU7 ELSE VDU6:PROCscreenman
```

## SMART COMPACTING REVISITED

In Issue 5 we carried a hint which automated the process of compacting, by repeatedly calling *COMPACT until the job was done. We have received two alternative ways of achieving the same end. Both have their merit in that they illustrate different techniques.

### *Graham Allan's Alias*

This technique defines an alias so that the routine will work regardless of which disc is in your machine, and can be called from environments other than Basic (e.g. Twin or Arthur). Here it is:

```
*SetMacro Alias$Compress Set Start$Ti
me <Sys$Time>|M Compact|M IF Sys$Time= S
tart$Time THEN Map ELSE Compress
```

To compact your disc, just type:

```
*Compress
```

Graham warns us that the method is not foolproof, however. If the number of compactions is very large, you may get an "Expansion too complex" error. In that case, just issue *Compress again.

### *Nic van Someren's SYS Call*

Nic's solution is to use a SYS call which returns both the total free space on a disc, and the size of the largest free block. He uses a short sequence of code to issue *Compact until the two are equal:

```
10 INPUT "Drive no/Disc name", D$
20 REPEAT
30 OSCLI("COMPACT "+D$)
40 SYS "ADFS_FreeSpace",D$ TO A%,B%
50 UNTIL A%=B%
```

## PART SCREEN TEMPORARY CACHE

By Barry Christie

It is often useful to be able to temporarily save all or part of the screen. You might for example wish to display an information window or pop up menu, and then restore the screen when it is no longer required. This is all made extremely easy with the command *SGET, which can be used to save any part of the screen as a sprite. For example, the sequence:

```
MOVE 100,100:MOVE 200,200
*SGET name
```

will save a rectangle of screen (bottom left co-ordinate 100,100, top right co-ordinate 200,200) as a sprite named *name*. To restore this rectangle, use:

```
*SCHOOSE name
PLOT &ED,100,100
```

This selects the named sprite, and plots it at 100,100. The only thing to check on is that you have configured sufficient sprite space (use *CON.SPR.n where n is the number of 8 or 32K pages of sprite RAM reserved).

## NAMED DISCS

By Graham Allan

By using ***NameDisc <disc spec.> <name>** you can give each disc a name (as distinct from a title) of up to 10 characters. You can then use this to specify a particular disc regardless of which drive it is in. It is also useful for example when using automated saves and backups, to ensure that the destination disc is of the right type. For example, it could be used in conjunction with the Automatic Backup hint in Issue 6. RU

# RISC USER magazine

## MEMBERSHIP

RISC User is available only on subscription at the rates shown below. Full subscribers to RISC User may also take out a reduced rate subscription to BEEBUG (the magazine for the BBC micro and Master series).

*All subscriptions, including overseas, should be in pounds sterling. We will also accept payment by Connect, Access and Visa, and official UK orders are welcome.*

| RISC USER SUBSCRIPTION RATES | | RISC USER & BEEBUG |
|---|---|---|
| £14.50 | 1 year (10 issues) UK, BFPO, Ch.I | £23.00 |
| £20.00 | Rest of Europe & Eire | £33.00 |
| £25.00 | Middle East | £40.00 |
| £27.00 | Americas & Africa | £44.00 |
| £29.00 | Elsewhere | £48.00 |

### BACK ISSUES

*We intend to maintain stocks of back issues New subscribers can therefore obtain earlier copies to provide a complete set from Vol.1 Issue 1. Back issues cost £1.20 each. You should also include postage as shown:*

| Destination | UK, BFPO, Ch.Is | Europe plus Eire | Elsewhere |
|---|---|---|---|
| First Issue | 40p | 75p | £2 |
| Each subsequent Issue | 20p | 45p | 85p |

## MAGAZINE DISC

*The programs from each issue of RISC User are available on a monthly 3.5" disc. This will be available to order, or you may take out a subscription to ensure that the disc arrives at the same time as the magazine. The first issue (with six programs and animated graphics demo) is at the special low price of £3.75. The disc for each issue contains all the programs from the magazine, together with a number of additional items by way of demonstration, all at the standard rate of £4.75.*

| MAGAZINE DISC PRICES | UK | Overseas |
|---|---|---|
| Single issue discs | £ 4.75 | £ 4.75 |
| Six months subscription | £25.50 | £30.00 |
| Twelve months subscription | £50.00 | £56.00 |

*Disc subscriptions include postage, but you should add 50p per disc for individual orders.*

*All orders, subscriptions and other correspondence should be addressed to:*

**RISC User, Dolphin Place, Holywell Hill, St Albans, Herts AL1 1EX.**
**Telephone: St Albans (0727) 40303**
*(24hrs answerphone service for payment by Connect, Access or Visa card)*